VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a

informatiky

Připojení grafického LCD panelu k zobrazovací jednotce

Microcomputer with Graphical LCD Display

2010/2011

Dalibor Kloss

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Dalibor Kloss**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Připojení grafického LCD panelu k zobrazovací jednotce
Microcomputer with Graphical LCD Display

Zásady pro vypracování:

Vyberte vhodný typ grafického LCD panelu černobílého i barevného, s vlastním řadičem. Navrhněte připojení k mikroprocesoru. Navrhněte a realizujte programové rozhraní pro grafické i textové ovládání LCD panelu.

1. Vyberte vhodné LCD panely s vlastním řadiče.
2. Navrhněte připojení LCD panelu k vybranému mikrokontroléru.
3. Navrhněte programové rozhraní pro ovládání LCD panelu v jazyce C.
4. Realizujte navržené rozhraní pro oba panely.
5. Otestujte rychlost, spolehlivost a zhodnoťte univerzálnost navrženého řešení.

Seznam doporučené odborné literatury:

http://www.microchip.com
http://www.atmel.com
http://zefiryn.tme.pl/dok/a06/pt02432t-a401.pdf
http://zefiryn.tme.pl/dok/a06/prg2412a-series-1.pdf
http://www.palmtech.com.tw/

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Petr Olivka**

Datum zadání: 19.11.2010

Datum odevzdání: 06.05.2011

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašují, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne ……. Podpis autora …….

Poděkování

Děkuji vedoucímu mé bakalářské práce, panu Ing. Petru Olivkovi za odbornou pomoc a rady, které mi poskytl při psaní této práce.

Abstrakt

Tato bakalářská práce se zabývá připojením černobílého a barevného LCD displeje k mikropočítači. V práci je popsán teoretický princip jednotlivých typů displejů a vybraného mikroprocesoru. Tyto displeje a mikroprocesor jsou nejprve popsány teoreticky, jsou zde popsané základní vlastností daných displejů a jejich řadičů. Pak následuje praktická realizace zapojených displejů s mikroprocesorem, kde jsou popisovány základní postupy k oživení daných displejů a následné porovnání některých naměřených vlastností daných displejů.

Klíčová slova: LCD, mikroprocesor, grafický displej, programování mikroprocesoru, Bresenhamův algoritmus

Abstract

This thesis deals with connecting monochromatic and color LCD display with microcomputers. In project is describes the theoretical principles of various types of displays and the selected microprocessors. These displays and microprocessor are first described in theory, are described the basic properties of the displays and their controllers. Then followed the practical realization of displays connected to the microprocessor, which are described the basic procedures for the recovery of the displays and the subsequent comparison of some measured properties of the displays.

Key words: LCD, microprocessor, graphic display, microprocessor programming, Bresenham algorithm

Seznam použitých zkratek

JDM – Jeans Dyekjar Madsen, navrhl universální programátor PIC

LCD – Liquid Crystal Display, technologie založená na tekutých krystalech

PIC – mikropočítač společnosti Microchip

PWM – pulsně šířková modulace

TFT – Thin Film Transistor, technologie pro barevné LCD

USART – zařízení pro sériovou komunikaci

Seznam použitých obrázků

Seznam tabulek

Obsah

1. Úvod

Od té doby, kdy bylo upuštěno od děrných pásek a štítků jako jediných prostředků využívaných pro dorozumívání člověka s počítačem, je komunikace zprostředkovávána zejména klávesnicí a monitorem. Já se v této práci zaměřím na oblast práce s monitorem respektive s grafickým displejem, na kterém se nám zobrazují určité znaky. Mým úkolem bude popsat a vyzkoušet tuto komunikaci s mikroprocesorem, co je vše třeba udělat a nastavit abychom dostali náš požadovaný text či grafické znaky na náš displej.

1.1 Cíle

Cílem této bakalářské práce je s vybranými grafickými displeji, které obsahují vlastní řadiče navrhnout a realizovat příslušné propojení s mikroprocesorem. Realizace proběhne pouze na nepájivé desce plošných spojů. Mikroprocesor bude řídit činnost řadiče, bude obsahovat uložený text nebo grafické znaky, které se poté budou zobrazovat na konkrétním displeji. Při této práci budou zkoumány vlastností těchto řadičů. Hlavní důraz bude kladen na zobrazování textové části pro pozdější využití těchto displejů v praxi, jako například různé terminály. Výsledná implementace kódu bude provedena v programovacím jazyce C.

2. Výběr vhodného LCD displeje

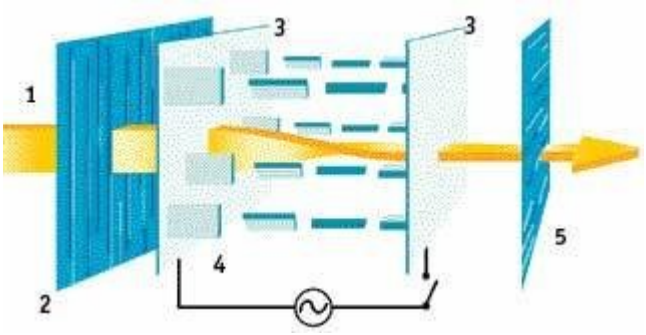
2.1 Popis technologii

2.1.1 Princip LCD

Princip LCD (Liquid Crystal Display) je založen, jak již z názvu vyplývá z technologie tekutých krystalů. Tekuté krystaly jsou látky, které se kromě tekutého a pevného stavu vyskytují také v tzv. kapalné krystalické fázi. V tomto stavu jsou tekuté, ale mají optické a elektromagnetické vlastnosti pevných látek.

Činnost LCD-displeje je založena na natáčení tekutých krystalů, z nichž jsou složeny jednotlivé obrazové buňky.

Pod tekutými krystaly svítí světlo, nejčastěji je to elektroluminiscenční výbojka. Její světlo buňka LCD buď nepropustí, utlumí nebo nechá projít k očím pozorovatele.



Obr. 1. – Polarizace pomoci tekutých krystalů

Na spodní a horní straně displeje jsou umístěny polarizátory (2, 5), ty propouštějí pouze světlo polarizované buď ve vodorovném, nebo svislém směru. Mezi těmito dvěma polarizačními filtry se nachází vrstva tekutých krystalů (3).

V průchozím stavu jsou tekuté krystaly buňky LCD šroubovitě pootočeny tak, že světlo (z podsvětlení) projde horizontálním polarizátorem, buňky pootočí, a světlo tak projde i druhým vertikálním polarizátorem. A jeden bod na displeji se rozzáří.

Druhým mezním stavem je situace, kdy světlo ze spodní strany displeje neprojde k očím pozorovatele. Toho se docílí tím, že se na elektrody tekutého krystalu připojí střídavé napětí. Tekuté krystaly se narovnají, spodní světlo projde prvním polarizátorem, ale krystaly je nepootočí, a tak je světlo druhým polarizátorem zastaveno. V tomto případě zůstane bod nerozsvícen [4].

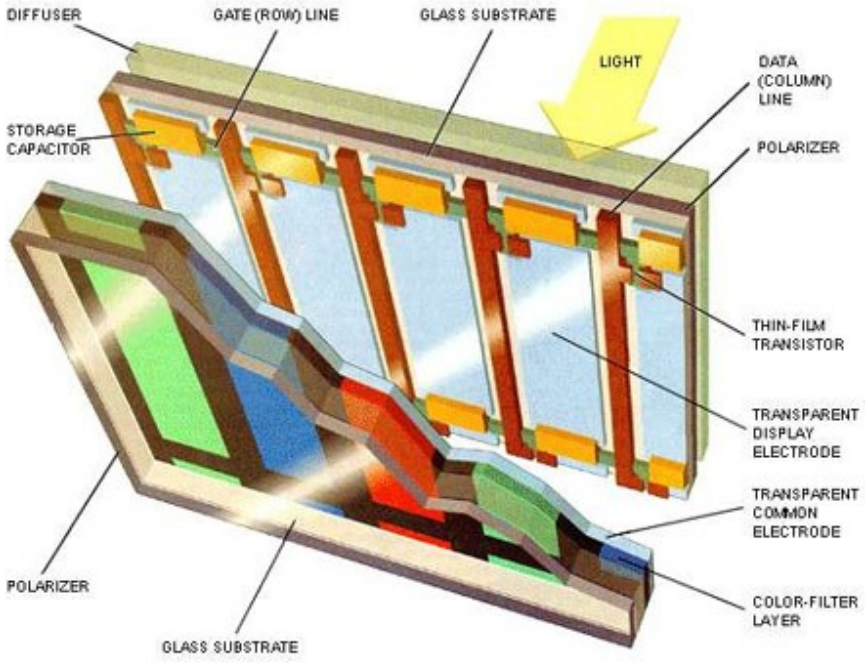
V praxi se samozřejmě nepoužívají pouze tyto dva mezní stavy (svítí, nesvítí) ale je zapotřebí zobrazovat body s různou intenzitou světla. Toho docílíme změnou velikosti střídavého napětí, připojeného k elektrodám tekutého krystalu.

2.1.2 Princip TFT

U této technologie je každý obrazový bod – pixel aktivně ovládán třemi tranzistory TF. Pro vznik obrazu potřebujeme dvě veličiny - světlo a barvu. Světlo je zajišťováno podsvětlujícími

lampami CCFL (Cold Cathod Fluorescent Lamp), které jsou intenzívním zdrojem primárního bílého světla. Výsledný obraz se z bílého světla získá technologií LCD (Liquid Crystal Display). Jakoukoliv barvu lze vytvořit ze tří barevných složek - červené, zelené a modré (RGB).

Každý obrazový bod je ohraničen dvěma polarizačními filtry, barevným filtrem (pro červenou, zelenou a modrou) a dvěma vyrovnávacími vrstvami. Vše je vymezeno tenkými skleněnými panely. Tranzistor každého obrazového bodu kontroluje velikost napětí, které prochází mezi vyrovnávacími vrstvami a elektrické pole působí na změnu struktury tekutého krystalu, čímž ovlivní natočení jeho částic. Tímto způsobem je možné regulovat několik desítek až stovek stavů tekutého krystalu, při kterých vzniká výsledný jas barevných odstínů. A protože se každý obrazový bod skládá ze tří základních barevných "sub-pixelů" (RGB), vznikají tak statisíce až miliony různých barevných odstínů.



Obr. 2. – TFT displej

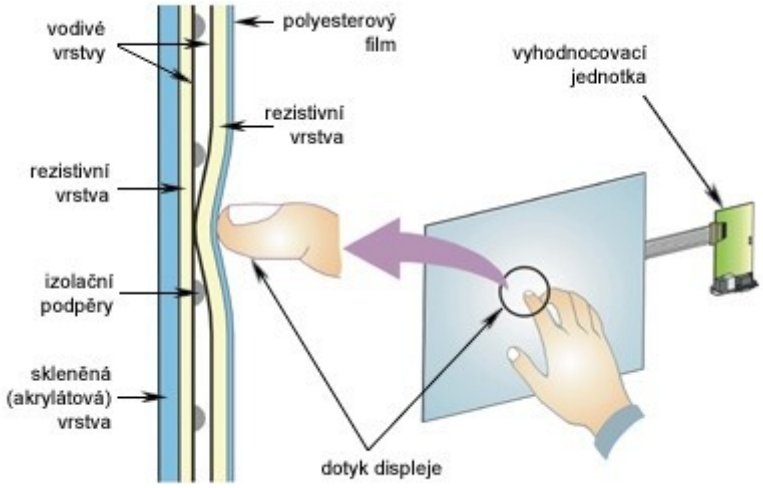
2.1.3 Princip dotykové obrazovky

Tato obrazovka nám dokáže detekovat přítomnost a místo doteku na zobrazovacím displeji.

2.1.3.1 Rezistivní technologie

Tento systém řešení je tvořen pružnou membránou, která je na povrchu displeje. Membrána je zevnitř pokryta velmi tenkou průhlednou kovovou vrstvou. Pod membránou je také vodivá průhledná vrstva, která je ale pevná. Mezi vrstvami je pak velmi tenká vzduchová mezera s rastrem izolačních podpěr, které vodivé vrstvy izolují od sebe. Obě vrstvy jsou připojeny k řadiči.

Při dotyku se horní vrstva prohne a v daném místě se vodivě spojí s vrstvou spodní. Mezi vrstvami pak začne procházet elektrický proud a řadič pak vypočítá na základě velikosti jednotlivých proudů polohu bodu dotyku. Princip technologie je znázorněn na následujícím obrázku.



Obr. 3. – Princip rezistivní technologie

Výhodou tohoto řešení displeje je především to, že k dotyku lze použít prakticky cokoli. Může to být špička prstu a třeba i v rukavici, tužka nebo jakýkoli jiný předmět. Jde tu v podstatě jen o vyvinutý tlak na horní vodivou vrstvu.

Rezistivní dotykové displeje jsou velice odolné a používají se proto například v průmyslových aplikacích.

2.1.3.2 Kapacitní technologie

Tato technologie je založená na vodivosti lidského těla. Povrch displeje je pokryt vodivou vrstvou. Při dotyku displeje prstem ruky vznikne mezi okraji displeje a vodivou rukou kapacita, přes kterou se uzavírá se elektrický obvod. Řadič pak analýzou vzniklých kapacit přesně určí polohu prstu.

Výhodnou vlastností tohoto systému je vysoká mechanická odolnost a také malá velmi nízká náchylnost na poruchy funkce vlivem ušpinění (mastnota, prach apod.). Velikou nevýhodou a omezením je to, že dotyk displeje funguje jen v případě, že se obrazovky dotýkáme elektricky vodivým předmětem. V případě, že displej budeme ovládat například rukou v kožené rukavici nebo s tyčinkou z plastu tak u tohoto displeje nám nebude displej reagovat.

Speciálním případem kapacitního displeje je pak tzv. projekční kapacitní displej. Využívá principu kapacitního displeje, ale s tím rozdílem, že vyzařuje elektrické pole do blízkého okolí. Pokud takový displej umístíme např. za nevodivou tenkou vrstvu skla nebo plexiskla, bude tento systém fungovat a bude přitom vysoce mechanicky odolný.

2.2 Nabídka černobílých displejů

Při výběru tohoto typu displeje jsme si nejdřív stanovili určité požadavky. V první řadě by měl být displej grafického typu, aby na něm bylo možné zobrazovat text a různé geometrické tvary. Displej by měl obsahovat i svůj řadič, který nám usnadní práci s tímto daným displejem a bylo by

dobré, kdyby daný displej obsahoval i znakovou sadu. Jako cenový limit jsme si stanovili částku pohybující se kolem 1000Kč.

Zde uvádím pouze některé varianty displejů, které byly v období září až listopad, tedy kdy se tento displej vybíral v aktuálních nabídkách příslušných obchodů. Uvádím zde 4 různé varianty displejů, z kterých jsem si vybíral, u každé této varianty mám vypsané základní vlastnosti příslušného displeje, cenou a dostupnost, která byla v období výběru toho to displeje.

Varianta 1:
LGM2412ASL
- prodejna: TME
- velikost: 240x128 bodů
- řadič: T6963C
- cena: 1212 Kč
- dostupnost: ano

Varianta 2:
LGM12064ESL
- prodejna: TME
- velikost: 240x64 bodů
- řadič: T6963C
- cena: 1455Kč
- dostupnost: ano

Varianta 3:
ATM12864D-FL
- prodejna: GME
- velikost: 128x64 bodů
- řadič: S6B0107
- cena: 550Kč
- dostupnost: ano

Varianta 4:
MG12864A-SBC/H
- prodejna: GME
- velikost: 128x64 bodů
- řadič: S6B0107
- cena: 423Kč
- dostupnost: ne

Při porovnávaní těchto variant displejů, tak pro variantu 4 vychází lepší cena než u varianty 3, i když oba displeje mají skoro totožné vlastnosti. Rozdíl mezi nimi je pouze v podsvětlení displeje, kde u varianty 4 má barvu modrou a u varianty 3 má barvu žluto-zelenou. Bohužel varianta 4 nebyla v té době na skladě obchodů. Existovala sice možnost objednání toho to displeje, ale po konzultaci s vedoucím bakalářské práce jsme se ještě rozhodli hledat další displeje, abychom nemuseli na tento displej čekat než by případně dorazil. Z tohoto důvodu se stávala tato varianta spíše jako nouzová.

Další nalezené displeje jsem uvedl do varianty 1 a 2. Porovnání těchto dvou variant, které se od sebe odlišují jak ve velikosti displeje, kde větší rozměr má varianta 1, tak i v ceně, která je příznivější taktéž pro variantu 1. Proto pro pozdější rozhodování neměla varianta 2 pro mne velký význam.

Z těchto důvodů se ve výsledku rozhodují pouze mezi variantou 1 a 3. Po následné konzultaci s vedoucím bakalářské práce jsme se rozhodli pro variantu 1, která je díky svému většímu displeji přijatelnější i pro pozdější případné využití v praxi.

2.3 Nabídka barevných displejů

U těchto displejů byly podmínky pro pořízení displeje totožné jako u černobílého (viz kapitola 2.2). Dostupnost barevných displejů na našem trhu je však výrazně nižší než nabídka černobílých, kde většina barevných displejů je pouze na objednávku.

Varianta 1:
3,2“ TFT 320QVT - prodejna: Ebay
- rozlišení: 320x240 bodů
- řadič: HX8347-A
- cena: cca 442 Kč (26 $)

Varianta 2:
3,2“ TFT LCD Module Display- prodejna: Ebay
- rozlišení: 400x240 bodů
- řadič: ILI9327
- cena: cca 578 Kč (34 $)

Varianta 3:
3,5“ GFT035EA320240Y - prodejna: TME
- rozlišení: 320x240 bodů
- řadič: HX8238-A
- cena: 1210 Kč
- dostupnost: ano

Varianta 4:
2,8“ PT0282432 - prodejna: TME
- rozlišení: 320x240 bodů
- řadič: HX8347-A
- cena: 846 Kč
- dostupnost: ano

První rozhodování, který displej si pořídit se vztahovalo na variantu 3 a 4. Kde oba tyto displeje byly na skladu obchodu, byly tedy dostupné. Varianta 4 se jevila díky nižší ceně a lepší zpracované dokumentace k řadiči přijatelněji než varianta 3. Proto jsme se po konzultaci rozhodli o pořízení tohoto displeje i s jeho konektorem, který jsme plánovali napájet na desku plošného spoje a z kterého bychom potom vyvedli konektory, které bychom připojili k mikroprocesoru. Bohužel jsme však při kontrole přijetí zásilky z obchodu zjistili, že daný displej, který nám byl poslán neodpovídá našemu vybranému displeji a proto jsme tento displej vrátili zpět.

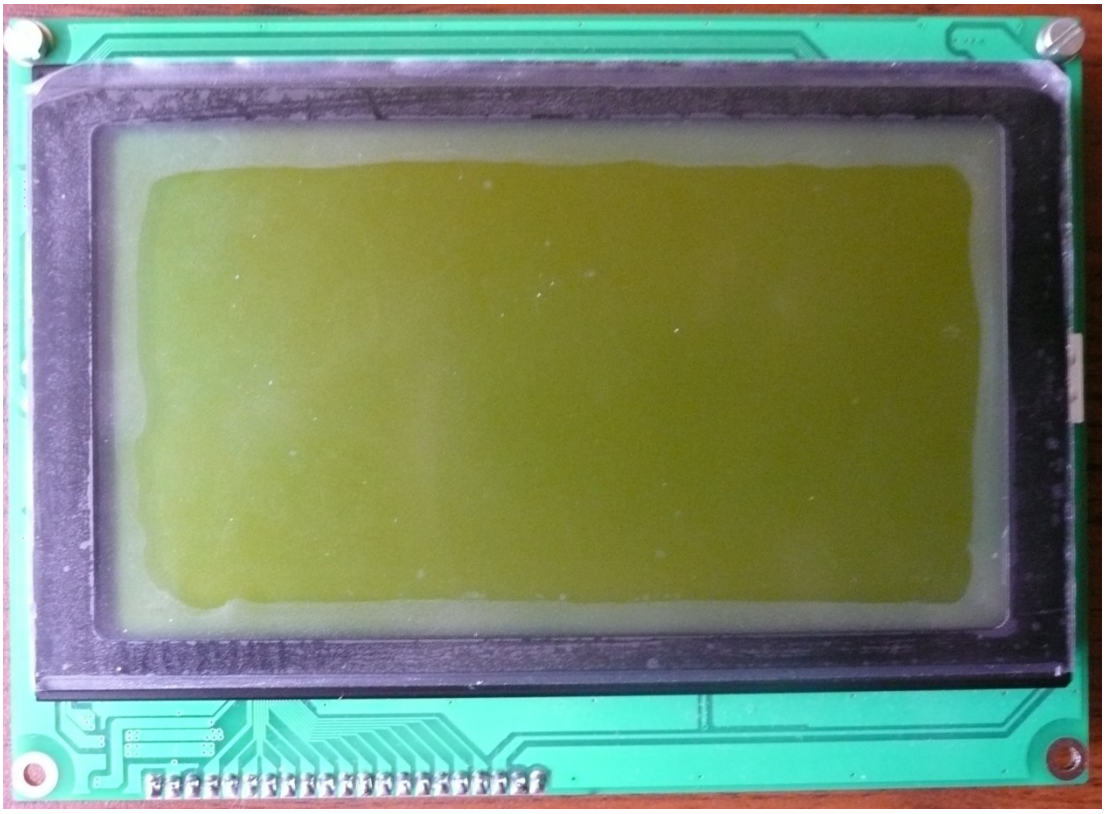
Následně, po této ne příliš šťastné situaci, s kterou jsme pouze ztratili čas, jsme museli znovu hledat nový displej, který by byl vhodný pro tuto práci. Od této chvíle jsem začal vybírat daný displej i v internetovém prostředí Ebay, v kterém je o mnoho větší výběr displejů i za přijatelnější cenu než kdekoliv u nás v obchodě.

Proto jsem i v tomto přehledu variant uvedl tyto displeje, z kterých jsem vybíral, jedná se o variantu 1 a 2. U obou těchto displejů byla přibližná cena přepočítána s kurzem 17Kč za 1 $, uváděna cena je i s dopravou, respektive doprava byla uváděna jako bezplatná. Navíc oba tyto displeje jsou vybaveny i dotykovou obrazovkou a vzorovými příklady kódu.

V rozhodování, který z těchto dvou displejů si pořídit, jsem přihlížel hlavně na kvalitu přikládaných dokumentací ohledně řadiče a displeje, a hlavně zda jsou v anglickém jazyce a jestli je zde vše potřebné popsané, hlavně vývody displeje, abych věděl, co mám na který pin připojit. Po konzultaci s vedoucím bakalářské práce jsme se rozhodli o pořízení displeje ve variantě 1, který má lepší cenu ale hlavně má lépe zpracovanou dokumentaci.

2.4 Černobílý grafický displej

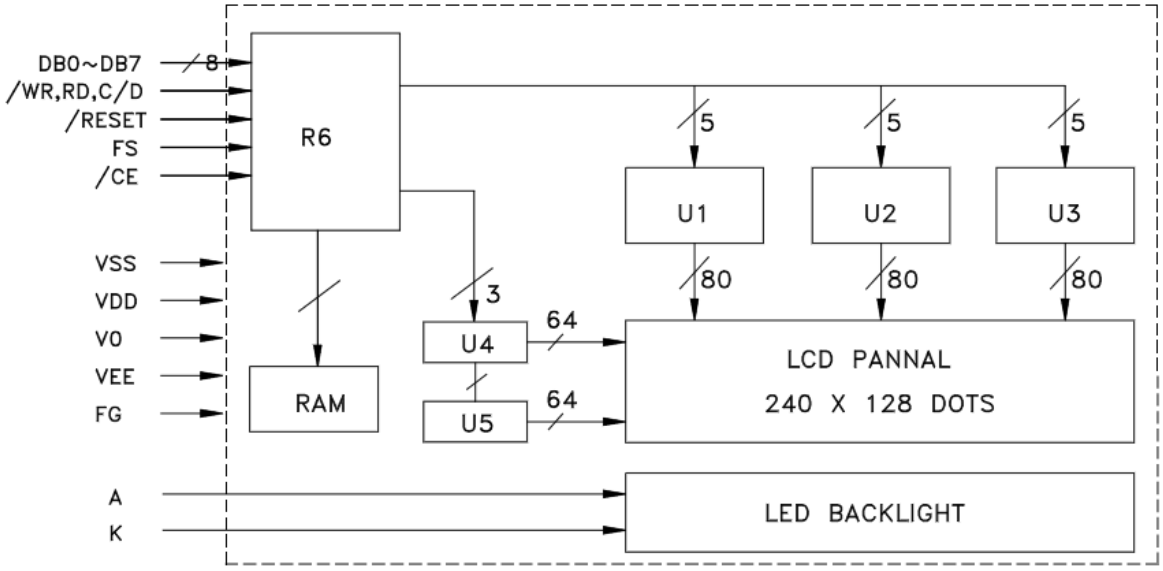
Pro práci byl zvolen tento černobílý grafický displej PQG2412AW-SYL vyrobeny firmou PALM TECHNOLOGY CO., LTD, jež patří mezi největší výrobce těchto displejů.



Obr. 4. – Černobílý grafický displej

Parametry tohoto displeje jsou:

- Rozlišení: 240x128 bodů
- Podsvícení: žluto-zelená
- Rozměry (šířka x výška x hloubka): 144 x 104 x 13,2 mm
- Viditelná plocha (šířka x výška): 114 x 64 mm
- Řadič: T6963C
- Datová sběrnice: 8 bitů



Obr. 5. – Blokové schéma černobílého displeje

2.4.1 Řadič T6963C

Výhoda tohoto displeje je, že obsahuje řadič T6963C od společnosti Toshiba, který patří mezi nejoblíbenější a hodně využívané u těchto displejů. Tento řadič dokáže řídit displej od velikostí 128 x 64 bodů až po velikost 240 x 128 bodů a lze ho jednoduše připojit k mikroprocesoru pomoci 8 - bitové datové sběrnice.

2.4.1.1 Paměť

Řadič obsahuje svou vlastní RAM paměť o velikosti až 64KB. Tuto paměť lze ještě rozdělit na oblast, kde se bude ukládat textová část, oblast pro grafickou část a oblast kde si může uživatel sám vytvořit své znaky.

2.4.1.2 Znaková sada

Řadič má v sobě uložené i znaky z ASCII tabulky, díky nimž lze psát text bez toho, abychom si pracně sami vytvářeli jednotlivé znaky.

2.5 Barevný grafický displej

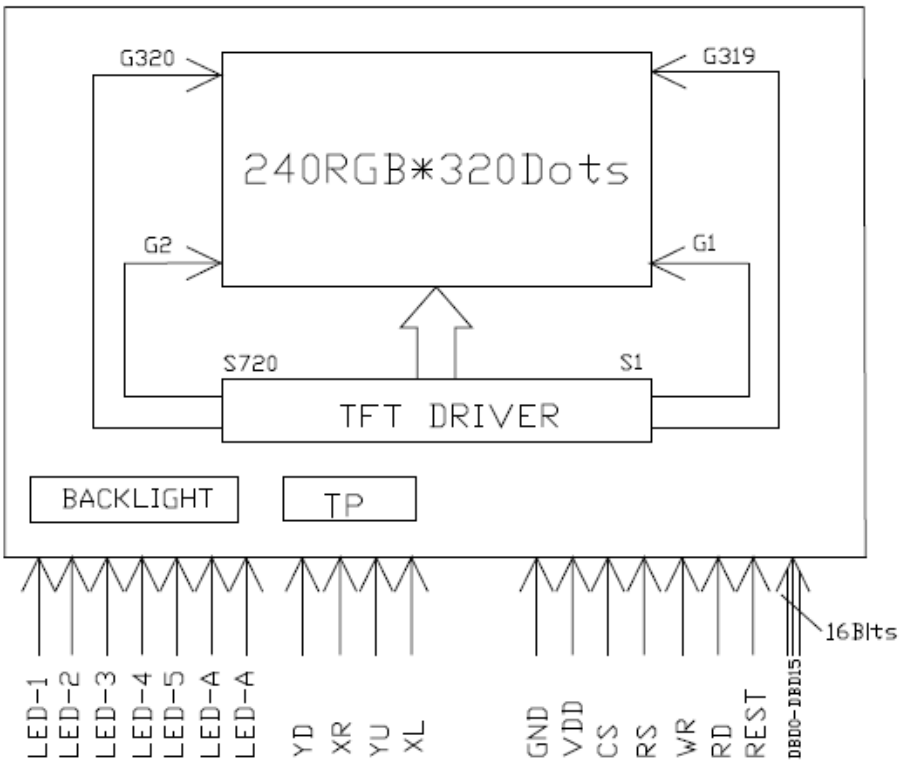
Jako další displej pro tuto práci byl zvolen barevný displej TFT 320QVT, který obsahuje i dotykový panel, který ale nebyl implementován, protože není součásti zadání této bakalářské práce a taktéž obsahuje čtečku paměťových karet, která taktéž nebyla implementována.

Rozlišení tohoto displeje je 240x320 bodů, maximální počet barev je 65 tisíc. Tento displej se používá nejčastěji v malých PDA a digitálních mobilních telefonech.

Obr. 6. – Barevný grafický displej

Parametry displeje:

- Rozlišení: 240x320 bodů
- Počet barev: 65 tisíc
- Řadič: HX8347-A
- Rozměry (šířka x výška x tloušťka): 57,54 x 79,2 x 4,40 mm
- Viditelná plocha (šířka x výška): 48,6 x 64,8 mm
- Podsvícení: bílé



Obr. 7. – Blokové schéma displeje

2.5.1 Řadič HX8347-A

2.5.1.1 Módy

Řadič tohoto displeje umožňuje zobrazit dva různé barevné módy:

- Normální barevný mód – počet barev je maximálně 65 tisíc
- „líný“ mód – počet barev je pouze 8

Řadič umožňuje komunikaci s mikroprocesorem po 8 - / 16 - / 18 - bitové datové sběrnici.

2.5.1.2 Napájení

Velikost vstupního napětí pro tento displej je +5V, přičemž řadič pracuje na napětí 2,3-3,3V, které je získáno vnitřním zapojením, a použitých usměrňovačů. Pro podsvětlení displeje je však doporučeno přidat pro toto napájení sériově zapojený rezistor o hodnotě 150 - 200Ω, protože maximální napětí pro podsvětlení je 3,2 V. V mém případě, když jsem zařadil rezistor o hodnotě 200 Ω, tak jsem dostal pro podsvětlení napětí 2,7V.

3. Připojení LCD k mikropočítači

3.1 Mikroprocesor

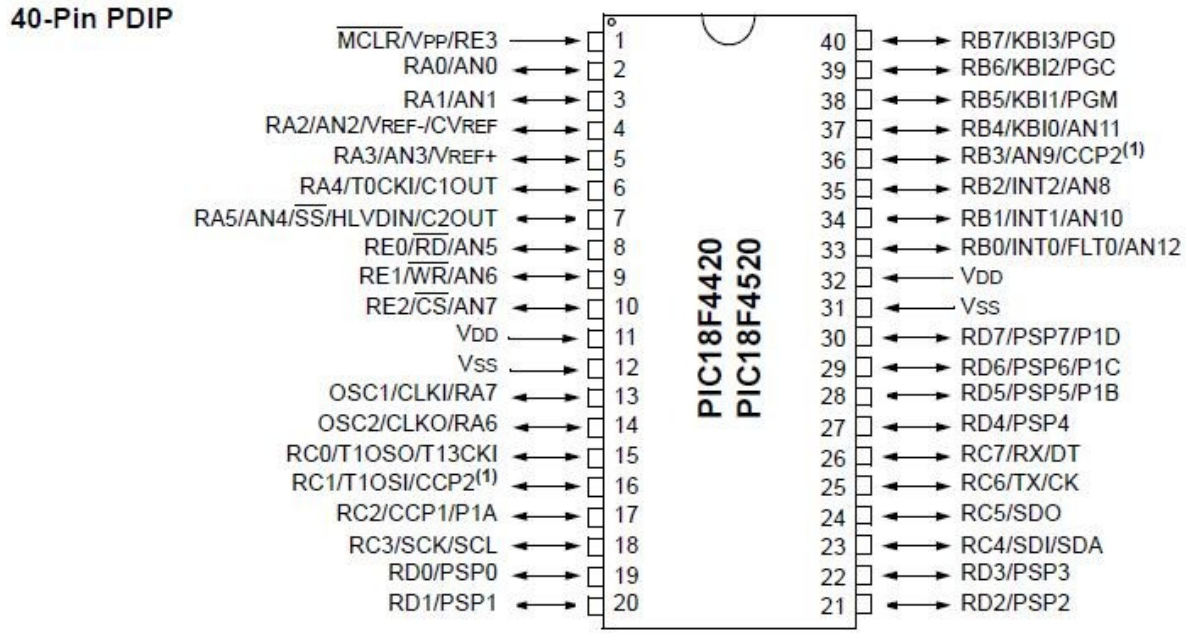
Pro toto téma bakalářské práce jsem si vybral mikroprocesor od Americké společnosti Microchip, která se zabývá výrobou jednočipových mikropočítačů. Tyto mikroprocesory jsou založené na harvardské architektuře, tedy mají oddělený paměťový prostor pro data a pro program. V mém případě se jedná o produkt řady PIC 18, přesněji o PIC18F452.

3.1.1 PIC

Zkratka PIC pochází z anglického jazyka a znamená Peripheral Interface Controller. Jak je z anglického názvu zřejmé, jde o mikroprocesor se zaměřením pro realizaci měřících a akčních členů řídicích systémů [5]. Akčním členem mám na mysli například stykač nebo vinutí krokového motoru a podobně.

3.1.2 Využití

Použití těchto procesorů se vyznačuje značnou variabilitou možných aplikací a především nízkou spotřebou a cenou. Ta se pohybuje řádově v desítkách až stovkách korun. Tyto důvody umožňují použití procesoru prakticky všude tam, kde je potřeba za velmi nízké náklady elektrická data nebo veličiny získat, přenést nebo zpracovat. V současné době je na trhu celá řada mikroprocesorů PIC s různou šířkou instrukčního slova nebo s různým vybavením periferií, které jsou implementovány přímo na čipu procesoru, například sériové rozhraní USART nebo výstup pulzně šířkovou modulaci PWM a mnoho dalších.



Obr. 8. – Popis vývodů mikroprocesoru

3.1.3 Vlastnosti

Jak již jsem uvedl dříve, pro mou práci jsem si vybral mikropočítač PIC18F452. Tento mikropočítač se řadí do rodiny 8 - bitových mikroprocesorů. Mezi jeho hlavní výhodu lze zařadit

jeho počet pinů, který je roven 40, díky nimž lze mikroprocesor využívat v mnoha oblastech, kde potřebujeme více vývodů pro svou práci. Obsahuje 5 vstupních/výstupních portů (A, B, C, D, E).

Frekvenci mikroprocesoru můžeme nastavovat s použitím externího krystalu až do hodnoty 40MHz, nebo můžeme využít vnitřní oscilátor, který lze nastavovat od hodnoty 31kHz až po maximální hodnotu 8MHz.

Jak již vyplývá z principu harvardské architektury, tedy kde je paměť rozdělena na dvě části, jedna část je vyhrazena pro program a druhá část je vyhrazená pro data, tak v našem případě obsahuje mikroprocesor paměť pro data o velikosti 1536 Byte, EEPROM paměť pro data o velikosti 256 Byte, a pro program paměť o velikosti 32768 Byte.

3.2 Připojení černobílého displeje k mikroprocesoru

Komunikace LCD displeje s mikroprocesorem probíhá po 8 - bitové datové sběrnici a 6 řídicími signály. Piny mikroprocesoru jsou připojeny přímo na piny displeje, tj. metoda pin to pin.

Datovou sběrnicí mám zapojenou na port D mikroprocesoru a řídicí signály na port B, viz následující tabulka. Mezi řídící signály patří piny LCD které jsou označené WR, RD, CE, C/D, RST. Jejich význam je následující:

Pin LCD	Symbol	Úroveň	Popis	Pin mikroprocesoru
1	FG	0 V	zem	-
2	GND	0 V	zem	-
3	VDD	5 V	napájecí napětí	-
4	V0	-	volitelné napětí pro displej	-
5	/WR	H/L	signál, jímž aktivujeme zápis dat do řadiče	RB2
6	/RD	H/L	signál, jímž aktivujeme čtení dat z řadiče	RB1
7	/CE	H/L	signál pro výběr displeje	RB3
8	C/D	H/L	signál pro přepínání mezi režimem instrukce nebo data	RB0
9	/RST	H/L	signál pro reset	RB4
10	DB0	H/L	datové vodiče	RD0
11	DB1			RD1
12	DB2			RD2
13	DB3			RD3
14	DB4			RD4
15	DB5			RD5
16	DB6			RD6
17	DB7			RD7
18	FS	H/L	signál pro výběr fontů	RB5
19	VEE	-10 V	záporné napětí	-
20	A	5 V	anoda podsvícení	-
21	K	0 V	katoda podsvícení	-

Tabulka 1 – Popis vývodů černobílého displeje

3.3 Připojení barevného displeje k mikroprocesoru

S barevným displejem lze komunikovat pouze po 8 nebo 16 – bitové datové sběrnici, z důvodů omezeného počtu vývodů displeje není zde podporována komunikace po 18 – bitové datové sběrnice, i když ji řadič podporuje. Je tedy na nás jaký typ komunikace si vybereme. Při výběru komunikace po 8 – bitové datové sběrnice využíváme datové vodiče DB0-7, ostatní DB8-15 musíme uzemnit, aby nedocházelo k rušení námi posílaných signálů. Při komunikace po 16 – bitové datové sběrnici využíváme všech datových vodičů.

V mém případě jsem se rozhodl využít pro komunikaci mezi mikroprocesorem a barevným displejem pomoci 16 – bitové datové sběrnice, která je zapojená na port D a port C mikropočítače. Další vodiče, které slouží k řízení tohoto řadiče, jsou připojeny na port B mikropočítače. Všechny vodiče jsou přímo propojené s mikroprocesorem a displejem, tj. metodou pin to pin. Jednotlivý popis vývodů na displeji je následující:

Pin LCD	Symbol	Napěťová úroveň	Popis	Pin mikroprocesoru
1	GND	0	zem	-
2	VCC	+5 V	napětí pro displej	-
3	NC	-	nevyužito	-
4	RS	H/L	signál pro reset řadiče	RB3
5	WR	H/L	signál pro zápis data/instrukce	RB1
6	RD	H/L	signál pro čtení dat	RB0
7 - 14	DB8 – DB15	H/L	datová sběrnice	RD0-7
15	CS	H/L	signál pro aktivování řadiče HX8347	RB2
16	F_CS	H/L	signál pro výběr	-
17	REST	H/L	signál pro reset řadiče	RB4
18	NC	-	nevyužito	-
19	LED-A	+5 V (rezistor 150Ω)	napětí pro podsvětlení displeje	-
20	NC	-	nevyužito	-
21 - 28	DB0 – DB7	H/L	datová sběrnice	RC0-7
29 - 34	D_	H/L	signály pro dotykový displej	-
35 - 40	SD	H/L	signály pro řízení čtečky paměťové karty	-

Tabulka 2 – Popis vývodů displeje

3.4 Programátor JDM

Na programování mikroprocesoru PIC18F452 jsem použil svůj vyrobeny programátor na základním principu JDM s kterým lze programovat mikroprocesory firmy Michrochip. Hlavní výhoda tohoto programátoru je, že mikroprocesor nemusíme složitě vyndávat ze svého slotu, ale můžeme přímo programovat zapojený v obvodu. Takovéto programování se nazývá „sériové

programování v zapojení“ nebo taky zkráceně ICSP. Programátor je připojený k počítači přes sériový port (RS232).

Sériový port na straně PC			
pin	**název**	**směr toku dat**	**popis**
1	CD	do PC	detekce zapojení
2	RxD	do PC	přijímá data
3	TxD	ven z PC	vysílá data
4	DTR	ven z PC	informuje, že kanál je připraven
5	GND		zem
6	DSR	do PC	informuje, že kanál je připraven
7	RTS	ven z PC	požadavek k přenosu
8	CTS	do PC	odpověď, aby zahájil přenos
9	RI	do PC	indikátor vyzvánění

Tabulka 3 – Sériový port

Programátor pro svou činnost využívá hlavně 3 signály, signál RTS, který slouží jako zdroj taktovacích signálů, DTR po kterém vysílá data a signál TxD s kterým zapíná programování.

Abychom mohli využit tento programátor, tak nesmíme zapomenout připojit blokovací kondenzátor mezi V_{CC} a GND a jehož hodnota by neměla být větší než 100µF. Při programování by neměl odběr proudu u ostatních obvodů přesáhnout 10 mA.

Vývod mikroprocesoru MCLR/V_{PP} nesmí být zapojený přímo na V_{CC}, ale musí být oddělen od tohoto napětí rezistorem větším než 1 kΩ a diodou, která zabrání aby se při programování nedostalo programovací napětí V_{PP} do napájecího napětí mikroprocesoru V_{CC}. V mém případě jsem zde zařadil rezistor o hodnotě 10 kΩ.

Od základní verze tohoto programátoru jsem provedl několik změn podle svých představ a požadavků. Nepoužívám zde napětí pro programování ze sériového portu, které se běžně pohybuje u stolních počítačů kolem 12V, které je nutné při programování mikroprocesoru, z důvodu abych zbytečně nenamáhal tento port, a z důvodu, že na každém počítači může být toto napětí jiné a hrozilo by, že programátor by nemusel fungovat správně na všech. Z toho to důvodu jsem se rozhodl, že využijí externí napěťový zdroj, na kterém si nastavím napětí 16V a pomoci napěťových usměrňovačů 7812 a 7805 si vytvořím klasické napětí pro napájení mikroprocesoru, displeje a potřebné programové napětí V_{PP} pro programování mikroprocesoru.

4. Popis základních vlastností

4.1 Černobílý grafický displej

4.1.1 Nastavení počátečních adres

Pro zahájení komunikace s mikroprocesorem si musíme nejdříve rozvrhnout rozložení RAM paměti řadiče, musíme zvolit počáteční adresu paměti, od které se budou ukládat textová data, a počáteční adresu, od které se budou ukládat grafická data.

```
#define Text_home       0x0000
#define Graphic_home    0x2000
```

4.1.2 Velikost fontů

V dalším kroku si musíme zvolit velikost fontů, tento řadič podporuje velikost fontů 5x8, 6x8, 7x8, 8x8, můžeme si povšimnout, že výška fontů je pořád stejná, nelze měnit výšku písma. V našem případě displej podporuje pouze velikosti fontů 8x8 bodů nebo 6x8 bodů, to nastavujeme pomoci řídicího pinu označený na displeji FS (pro font 8x8 musíme nastavit 1 a pro font 6x8 musíme nastavit 0). Aby řadič zobrazoval data na správném místě displeje, musíme mu určit počet zobrazovaných znaků na jeden řádek, v mém případě když si zvolím font 8x8, tak počet znaků na jeden řádek je maximálně 30, pro font 6x8 je maximálně 40.

$$počet\ znaku\ na\ jeden\ řádek = \frac{240}{8(6)}$$

4.1.3 Posílání data řadiči

Jestliže chceme řadiči poslat nějakou instrukci, data, tak musíme nejprve zkontrolovat, jestli je řadič připraven provést náš požadavek. Tuto kontrolu jsem naprogramoval ve funkci nazvanou *CheckStatus*, jedná se zde o to, jestliže chceme zjistit stav řadiče, tak musíme nastavit na řídicí piny RD=0, WR=1, CE=0, C/D=1 a na datové sběrnici (DB0 – DB7) získáme stav řadiče:

STA7 D7	STA6 D6	STA5 D5	STA4 D4	STA3 D3	STA2 D2	STA1 D1	STA0 D0

STA0	ověření schopnosti vykonat příkaz	0: nepřipraven 1: připraven
STA1	ověření schopností čtení/zápisů dat	0: nepřipraven 1: připraven
STA2	ověření schopností čtení dat v auto režimu	0: nepřipraven 1: připraven
STA3	ověření schopností zápisu dat v auto režimu	0: nepřipraven 1: připraven
STA4	nevyužito	
STA5	ověření schopností práce řadiče	0: nepřipraven 1: připraven
STA6	příznak chyby	0: není chyba 1: chyba

STA7	ověření podmínky blikání	0: vypnuto 1: zapnuto

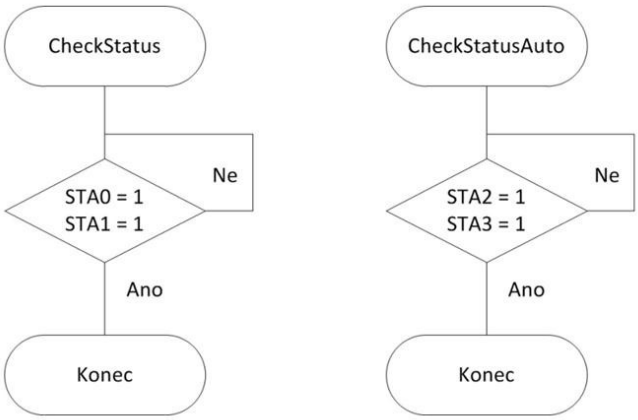
Tabulka 4 – Zjišťování stavu řadiče

Důležité jsou pro nás hlavně hodnoty v STA0 a STA1, pokud však nepoužíváme auto režim (viz dále) pro ukládání dat do paměti, v tomto případě bychom museli kontrolovat STA2 a STA3, to řeším ve funkci CheckStatusAuto.

Musíme počkat do té doby než v STA0 a STA1 bude v obou a ve stejný okamžik 1, jinak nám toto signalizuje, že řadič je zaneprázdněny a náš požadavek by nemusel správně splnit. STA0 nás informuje o tom, jestli je řadič připraven vykonat nějaký povel. STA1 nás informuje o tom, zda je řadič připraven k čtení nebo zápisů dat. Toto platí i analogický pro STA2 a STA3, kde v STA2 kontrolujeme připravenost pro automatické čtení dat a v STA3 připravenost pro zápis dat.

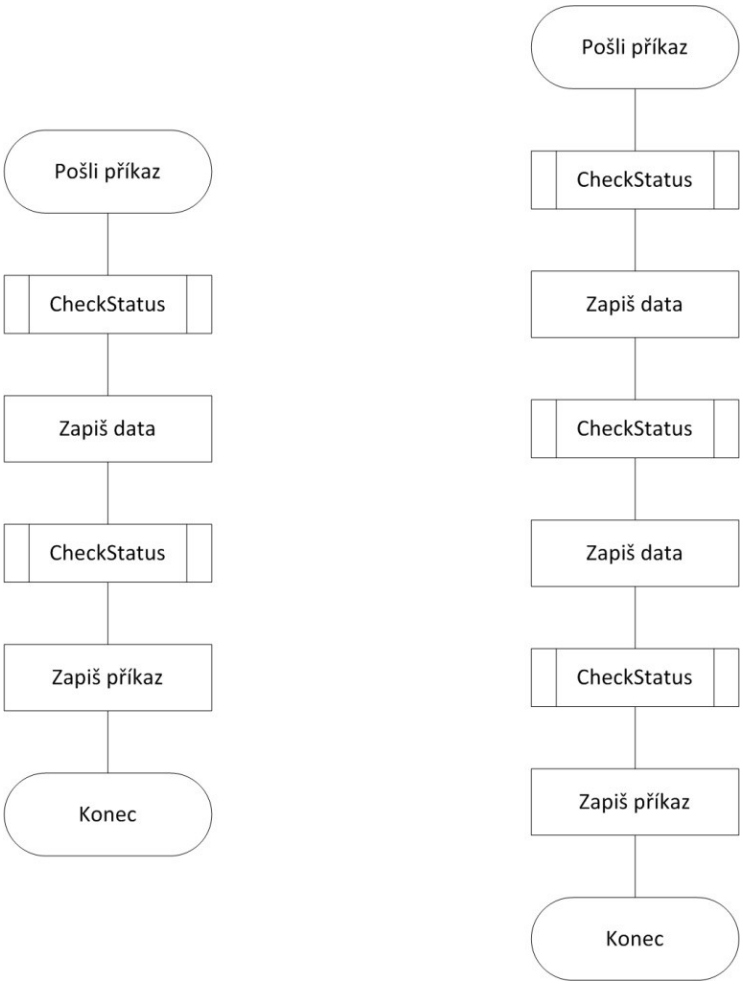
```
void CheckStatus (void)
{
 uint8_t tmp;
 TrisD = 0xff;
 do
 {
  LCD_RD = 0;
  LCD_CE = 0;
  tmp = LCD_Out;
  LCD_RD = 1;
  LCD_CE = 1;
 } while ((tmp & 0x03) != 0x03);
 TrisD = 0x00;

}
```



Obr. 9. – Vývojový diagram pro kontrolu řadiče

Při posílání dat řadiči, tak musíme nejprve provést kontrolu, viz výše, zda je řadič připraven, případně počkat až bude připraven a následně můžeme poslat data, v dalším kroku musíme opět zkontrolovat stav řadiče a nakonec pošleme příkaz, tento postup platí, jestliže posíláme jedny data, v případě potřeby zápisu dvou dat postupujeme obdobně, akorát mezi posílající data musíme přidat jednu kontrolu stavu řadiče, viz níže.



Obr. 10. – Vývojový diagram posílání dat

4.1.4 Příkazy pro řadič

Příkaz	Kód příkazu	Funkce
Nastavení registru	0010 0001	nastavení ukazatele pro kurzor
	0010 0010	nastavení offset registru
	0010 0100	nastavení ukazatele na adresu
Set control word	0100 0000	nastavení textové počáteční adresy
	0100 0001	nastavení počtu znaků na jeden řádek
	0100 0010	nastavení grafické počáteční adresy
	0100 0011	nastavení počtu znaků na jeden řádek
Mode set	1000 X000	OR mód
	1000 X001	EXOR mód
	1000 X011	AND mód
	1000 X100	text atribut mód
	1000 0XXX	vnitřní CG ROM mód
	1000 1XXX	vnější CG RAM mód
Display mode	1001 0000	vypnutí displeje
	1001 XX10	zapnutí kurzoru

	1001 XX11	zapnutí blikání kurzoru
	1001 01XX	zapnutí textového režimu, grafický režim je vypnut
	1001 10XX	zapnutí grafického režimu, textový režim je vypnut
	1001 11XX	zapnutí textového a grafického režimu
Nastavení kurzoru	1010 0000	kurzor je vysoký 1 pixel
	1010 0001	kurzor je vysoký 2 pixely
	1010 0010	kurzor je vysoký 3 pixely
	1010 0011	kurzor je vysoký 4 pixely
	1010 0100	kurzor je vysoký 5 pixelů
	1010 0101	kurzor je vysoký 6 pixelů
	1010 0110	kurzor je vysoký 7 pixelů
	1010 0111	kurzor je vysoký 8 pixelů
Data auto read/write	1011 0000	nastavení automatického zápisu
	1011 0001	nastavení automatického čtení
	1011 0010	zrušení auto režimu
Data read/write	1100 0000	zapíše data a zvýši ukazatel adresy (ADP)
	1100 0001	přečte data a zvýši ukazatel adresy (ADP)
	1100 0010	zapíše data a sníží ukazatel adresy (ADP)
	1100 0011	přečte data a sníží ukazatel adresy (ADP)
	1100 0100	zapíše data a s ukazatelem adresy neudělá nic
	1100 0101	přečte data a s ukazatelem adresy neudělá nic
Bit set/reset	1111 0XXX	bit reset
	1111 1XXX	bit set
	1111 X000	bit 0
	1111 X001	bit 1
	1111 X010	bit 2
	1111 X011	bit 3
	1111 X100	bit 4
	1111 X101	bit 5
	1111 X110	bit 6
	1111 X111	bit 7

Tabulka 5 – Příkazy řadiče

4.1.5 Rozdělení pamětí

Uživatel má k dispozici pro svou práci s displejem dvě oblasti kam může ukládat data, která se mají zobrazit na displeji. Jedná se o oblast textovou a grafickou, uživatel může tyto oblasti zobrazovat v různých módech například: OR nebo AND. Všechny oblasti tak mohou být zobrazeny najednou nebo jednotlivě. Nyní bude následovat série fotografií, na kterých zobrazím princip rozdělení paměti. Na každém obrázku mikroprocesor obsahuje tentýž zdrojový kód, měním pouze režim řadiče, odkud má čerpat data pro zobrazování.

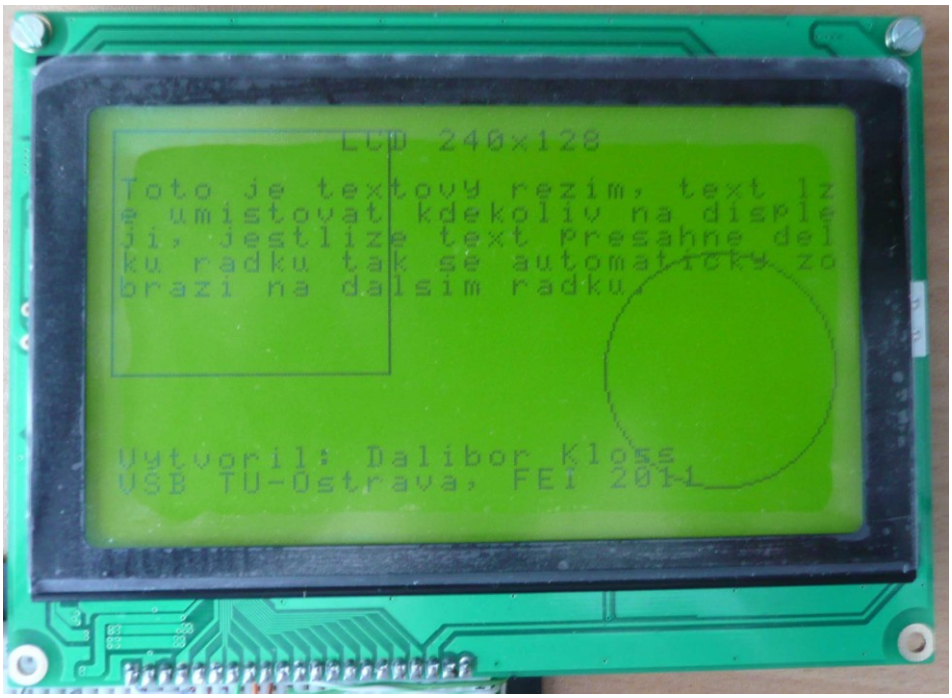
Následuje obrázek s nastavením řadičem pouze jako textový režim.

Obr. 11. – Textový režim

Nyní následuje obrázek kde je nastaven pouze grafický režim.

Obr. 12. – Grafický režim

A na posledním obrázku této série je zobrazen displej s výsledkem, jestliže jsme nastavili řadiči režim zobrazení OR. V této situaci řadič promítne na displej obsah jak textové tak i grafické paměti.



Obr. 13. – Režim OR

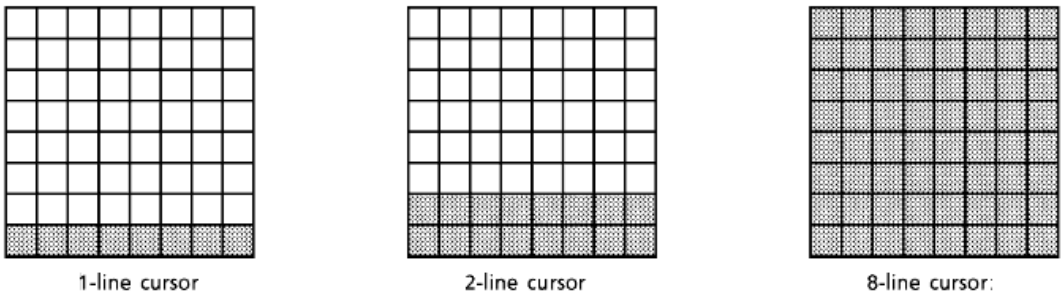
4.1.6 Nastavení adress pointerů

Address pointer je ukazatel na místo v RAM paměti displeje, představuje číslo, které nám ukazuje další volnou/obsazenou pozici v paměti. Jakmile zapisujeme/čteme data do/z paměti, například text, který chceme zobrazit na displeji, tak musíme vždy, když uložíme určitou hodnotu do paměti, zvednout/snížit ukazatel, tj. poslat řadiči příkaz DATA READ/WRITE pro zápis/čtení dat a zvýšení/snížení ukazatele adresy, aby nám ukazoval pro pozdější zápis na jiné místo v paměti, abychom si nepřepisovali původní hodnoty, které zde byly uloženy.

Této situaci se však můžeme vyhnout tím, že řadiči pošleme příkaz z kategorie DATA AUTO READ/WRITE nastavení auto režimu. V tomto případě, když budeme chtít zapisovat data do paměti, tak pošleme tento příkaz a řadič nám bude automatický zvedat hodnotu address pointeru.

4.1.7 Nastavení kurzorů

Umístění kurzoru specifikujeme pomoci X a Y adresy kdekoliv na obrazovce. Řadič nám taky dovoluje nastavení velikosti kurzoru a to od velikosti 1 pixelu až po velikost 8 pixelů. Toto nastavení se provádí příkazem z kategorie Nastavení kurzoru (viz tabulka příkazů). Můžeme však kurzor nastavit podle naši potřeby, aby nám klasický svítil nebo blikal.



Obr. 14. - Nastavení velikosti kurzoru

4.1.8 Nastavení offset registru

Offset registr se používá pro určování oblasti RAM externího znakového generátoru. Má 16 - bitovou adresní sběrnici, horních 5 bitů je určováno offset registrem, středních 8 bitů je určováno z kódu znaku a poslední 3 bity jsou určovány řádkovým čítačem.

offset registr					**kód znaku**								**řádkový čítač**		
ad15	ad14	ad13	ad12	ad11	ad10	ad9	ad8	ad7	ad6	ad5	ad4	ad3	ad2	ad1	ad0

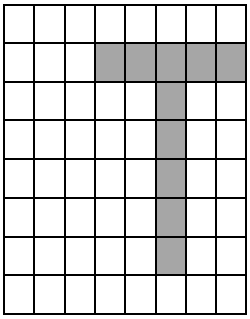
Tabulka 6 – Offset registr

Přehled mezi nastavením offset registru a uložením dat v RAM pamětí

offset registr	**oblast uložení dat v RAM (počátek - konec)**
00000	0000 – 07FF H
00001	0800 – 0FFF H
00010	1000 – 17FF H
11101	E800 – EFFF H
11110	F000 – F7FF H
11111	F800 – FFFF H

Tabulka 7 – Nastavení offset registru

Nastavením offset registru například na hodnotu 02 H (00010), kód znaku bude například 80 H, tak znakový generátor bude začínat od adresy 1400 H v paměti RAM.



adresa	**data**
1400 H	00 H
1401 H	1F H
1402 H	04 H
1403 H	04 H
1404 H	04 H
1405 H	04 H
1406 H	04 H
1407 H	00 H

Tabulka 8 – Tvorba znaku

4.1.9 Rolování textu

Jelikož samotný řadič displeje nemá v sobě funkci pro rolování textu, tak tuto hodně používanou a vyžadovanou funkci jsem naimplementoval sám. V podstatě se jedná o to, že jakmile zapisuji data do RAM paměti a nacházím se konci viditelných míst displeje, tj. na posledním řádku (15), tak jednoduše změním počátek viditelné plochy, o jeden řádek. To provedeme tak, že pošleme řadiči nový počátek pro textovou část a tím pádem se nám celé okno s textem posune o jeden řádek výš.

```
CheckStatus();
ZapisData((Text_home + ((cislo -1) * 0x001E)) & 0xff);
CheckStatus();
ZapisData((Text_home + ((cislo -1) * 0x001E)) >> 8);
CheckStatus();
ZapisPrikaz(Set_text_home_address);
```

4.2 Barevný grafický displej

4.2.1 Nastavení velikosti zobrazované plochy

Jako první data, která pošleme řadiči, by měla být informace o velikosti displeje. Z důvodu, aby řadič věděl, do jakých části pamětí bude mikropočítač zapisovat svá data. Toto nastavení se provádí v registrech 02 H až 09 H, a to v takovém pořadí, že si nejprve určíme adresu prvního sloupce (to odpovídá registru 02 H a 03 H), následuje adresa posledního sloupce (registr 04 H a 05 H) a taktéž adresa prvního řádků (registr 06 H a 07 H) a posledního řádků (registr 08 H a 09 H).

4.2.2 Umisťování textu na displeji

Umisťování a ukládání textu na obrazovce probíhá pomoci čítače adres, který je obsažen v řadiči. Pomoci tohoto čítače ukládáme/čteme data do/z pamětí podle adresy která je ve tvaru souřadnic. Tyto adresy však musí být v rozsahu pro hodnoty X od 0 do 239, pro hodnoty Y v rozsahu od 0 do 319.

Jakmile zapisujeme data do pamětí, tak tento čítač adres nám automatický přičítá nebo odečítá jedničku, podle toho jak máme nastaveny registr přístupu k paměti (registr 0x16), v tomto případě nám řadič prochází nebo ukládá data postupně od adresy [0,0] nebo postupuje z opačného konce tj. od [239,319] adresy. V tomto registru přístupu k paměti (registr 0x16) můžeme nastavovat nejenom směr refreshování obrazovky, ale můžeme zde volit i jaký použijeme barevný filtr pro obrazovku, jestli to bude klasický RGB nebo BGR, v němž by byly prohozené červená a modrá barva.

4.2.3 Otáčení displeje

Na rozdíl od černobílého displeje, který neumožňuje otáčení displeje, tak tady je tato funkce v řadiči implementována. Řadič nám tady dokonce dovoluje otáčení displeje o celých 360°, ale není se čemu divit, protože tento displej se hodně využívá, jak již zde bylo uvedeno v různých PDA a mobilních telefonech. Otáčení displeje se provádí opět nastavením v registru přístupu do paměti (registr 0x16).

Nastavení MY=0, MX=0, MV=0 registru 0x16

Obr. 15. – Otočení displeje o 0°

Nastavení MY=1, MX=0, MV=0 registru (0x16)

Obr. 16. – Otočení displeje o 270°

4.2.4 Rolování textu

Rolování textu je zde řešené řadičem. Jestliže požadujeme tuto funkci, tak musíme nejprve nastavit několik registrů.

Pro správné zobrazování textu, či grafiky je nutné nejprve řadiči určit velikost plochy kde se budou data zobrazovat. Pro tyto účely jsou zde následující registry, registry 0x0E až 0x15.

Rychlost rolování textu závisí na frekvenci řadiče displeje, která je při výchozím nastavení 5,5 MHz.

4.2.5 Znaková sada

Tento řadič bohužel neobsahuje v sobě žádnou uloženou znakovou sadu. Z toho to důvodu, jestliže chceme na displeji zobrazovat jakýkoliv text, tak si musíme nejprve připravit jednotlivé znaky, které si uložíme do pamětí mikroprocesoru a pak si je jednotlivě posíláme na displej.

Je třeba si uvědomit, že ve výpočetní technice lze jakýkoliv znak vyjádřit nějakou číselnou hodnotu v desítkové soustavě, viz ASCII tabulka. Pro nás je důležitá oblast tabulky od hodnoty 32, která představuje mezeru, až po hodnotu 126, což je tilda. Tyto znaky je třeba všechny si připravit a uložit do mikroprocesoru pro pozdější použití. Ostatní znaky z ASCII tabulky nejsou pro nás v tomto případě potřebné.

Pro tvorbu znaku je třeba se rozhodnout, jakou velikost bude mít daný znak, zda bude vysoký a široký 8 pixelů nebo víc či méně. Pro tento řadič jsem vytvořil dva druhy velikosti znaků. Pro normální text jsem zvolil výšku jednoho znaku 12 pixelů, pro nadpisy jsem zvolil dvojnásobek, tedy 24 pixelů.

Vytvořené znaky dané velikosti jsem uložil všechny do jednoho pole, toto pole je složeno z hexadecimálního vyjádření jednotlivých znaků (viz níže). Jakmile chci psát jakýkoliv text, tak napíšu požadovaný text a po jednotlivých znacích procházím tento řetězec a od každého odečtu hodnotu 32 a vyberu požadovaný znak z uloženého pole, neboť v uloženém poli nejsou obsažené hodnoty z ASCII tabulky do hodnoty 31.

Zobrazování uložených znaků na displej spočívá v tom, že řadiči budeme postupně posílat hexadecimální data, která představují náš vytvořený znak. Například pro můj vytvořený malý font, pro normální text, představuje pro jeden znak 12 hodnot hexadecimálních dat, viz obr. 17.

7	6	5	4	3	2	1	0	HEX hodnota
0	0	0	0	0	0	0	0	0x00
0	0	0	0	0	0	0	0	0x00
1	1	1	1	1	0	0	0	0xF8
0	1	0	0	0	0	0	0	0x40
0	1	0	1	0	0	0	0	0x50
0	1	1	1	0	0	0	0	0x70
0	1	0	1	0	0	0	0	0x50
0	1	0	0	0	0	0	0	0x40
0	1	0	0	0	0	0	0	0x40
1	1	1	0	0	0	0	0	0xE0
0	0	0	0	0	0	0	0	0x00
0	0	0	0	0	0	0	0	0x00

Obr. 17. – Vytvoření znaků

4.3 Zobrazovací algoritmy

Tyto algoritmy jsou použité jak u černobílého, tak i u barevného displeje.

4.3.1 Kreslení úsečky - Bresenhamův algoritmus

Pro kreslení úsečky využívám Bresenhamův algoritmus, jenž popisuje rasteriaci což je převod základních grafických objektů do systémů posloupnosti obrazových bodů. Toto je důležité při práci s různými výstupními zařízeními, které nejsou schopny zobrazit přesně daný objekt, ale při svém zobrazení využívají rastr.

Breansshamův algoritmus funguje na principu hledání nejbližších bodů, které leží nejblíže k dané úsečce.

Zadáme počáteční bod, jehož souřadnice jsou X1, Y1 a koncový bod se souřadnicemi X2, Y2 úsečky. Nyní provedeme rozdíl hodnot X a Y hodnot koncového bodu od počátečního, abychom zjistili, v jakých pozicích se body vůči sobě nachází, a podle toho pak budeme přičítat nebo odečítat 1, což nám bude představovat krok. Následně zobrazíme počáteční bod na displej. Pak je nutné vybrat si takzvanou řídící osu, dx nebo dy. V závislosti na vybrané ose vypočteme predikci.

- Predikce pro osu X se počítá podle vzorce 2 * dy – dx
- Predikce pro osu Y se počítá podle vzorce 2 * dx – dy

V závislosti na vybrané ose se spustí cyklus, dokud se x1 != x2, nebo dokud se y1 != y2, tzn. dokud není dosažen koncový bod. K hodnotě X1 se přičte hodnota kroku X, popřípadě se k hodnotě Y1 přičte hodnota kroku Y. Následně se vyhodnocuje predikce, je li větší než 0, k Y1 (popřípadě k X1) se přičte krok Y (popřípadě krok X) a vypočítá se nová predikce. Pro osu X se predikce (p) počítá následujícím vzorcem [8]:

$$p > 0 : p = p + 2 * dy - 2 * dx$$
$$p < 0 : p = p + 2 * dy$$

Pro osu Y se predikce (p) počítá následujícím vzorcem:

$$p > 0: p = p + 2 * dx - 2 * dy$$
$$p < 0: p = p + 2 * dx$$

Díky těmto znalostem můžeme snadno zobrazovat i obdélníky a čtverce.

5. Programové rozhraní

5.1 Společné ovládání

Z důvodů odlišných vlastností jednotlivých displejů, lze mít pro oba displeje stejnou pouze část grafického rozhraní, a to kreslení úsečky, obdélníků a kruhu. U obou displejů mám tyto funkce stejně pojmenované. V ostatních případech jsou sice některé metody podobné, ale nejsou naprosto stejné, protože displeje obsahují rozdílné řadiče a oba displeje jsou připojeny různým způsobem k mikroprocesoru, kde černobílý displej je propojen s mikroprocesorem po 8-bitové datové sběrnici a naproti tomu barevný displej je připojen pomoci 16-bitové datové sběrnice.

5.1.1 Společná část

První funkcí, s kterou se zde budu zabývat, je kreslení úsečky, její název je ***void usecka (int x1, int y1, int x2, int y2)***. Vstupní parametry nám určují počátek (x1, y1) a koncový bod úsečky (x2, y2). U barevného displeje je zde navíc jedna vstupní hodnota, s kterou zadáváme barvu úsečky, můžeme zadávat přímo číselnou hodnotu v hexadecimální soustavě, která nám vyjadřuje nějakou barvu, nebo můžeme využít mnou již připravené a pojmenované barvy, které jsou uložený v hlavičkovém souboru barevného displeje. V tomto algoritmu využívám znalosti z již zmíněného Bresenhamova aloritmu (kapitola 4.3.1).

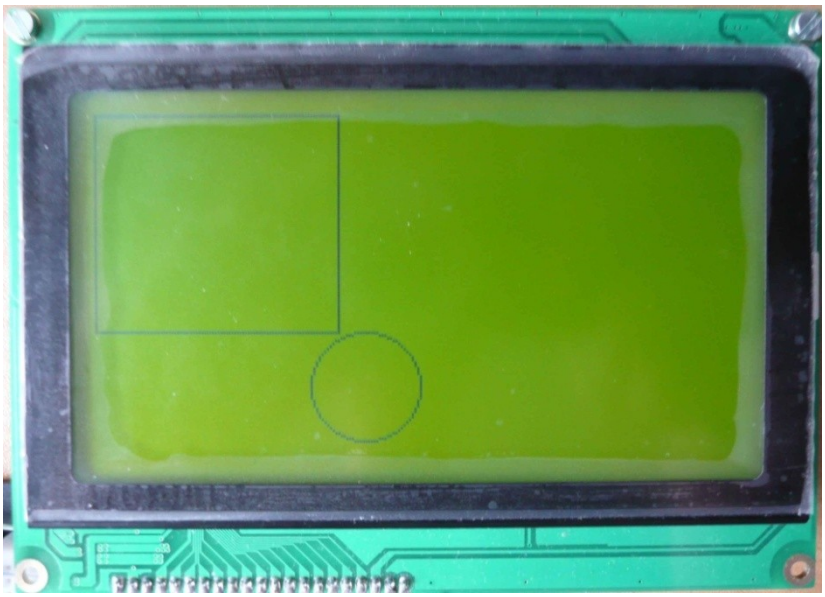
Další společnou funkci obou displejů je funkce, která slouží pro kreslení kružnice. Tato funkce je pojmenována ***void kruh(int x, int y, int r)***, u barevného je zde opět přidán jeden atribut a to pro zadání barvy. Kružnice se klasický definuji svým poloměrem a středem. Při výpočtu bodů na kružnici nás zajímá hlavně souřadnice oblouku 45 stupňů, neboť všechny ostatní body dostaneme záměnou souřadnic a změnou znamének. Princip vykreslení je opět na Bresenhamova algoritmu úsečky (viz kapitola 4.3.1).

V další metodě, která se zabývá kreslením obdélníků, využívám již zmíněnou funkci pro kreslení úseček. Daná metoda se nazývá ***void obdelnik (int x1, int y1, int x2, int y2)***, kde zadáváme za x1 a y1 souřadnice levého horního rohu, x2 a y2 nám popisuje pravý dolní roh, u barevného je zde opět vstupní parametr pro barvu. Princip této funkci je velmi snadný, nejdříve nakreslíme horní úsečku následně dolní a nakonec pravou a levou část obdélníků, viz následující kód barevného displeje:

```
usecka( x1, y1, x2, y1, color);      // horní část
usecka( x1, y2, x2, y2, color);      // dolní část
usecka( x2, y1, x2, y2, color);      // pravá část
usecka( x1, y1, x1, y2, color);      // levá část
```

Testovací metoda, která zde byla vytvořená pro testování grafických tvarů má název *grafickaOblast()*, v této metodě mám u obou displejů nastavené, aby řadič vykreslil obdélník s levým horním okrajem o souřadnicích [1,1] a pravý dolní okraj na souřadnicích [90,80]. Následující zobrazovaný tvar je kruh o středu na souřadnicích [100,100] a poloměrem 20 pixelů. Nyní přikládám výsledný obraz displeje s těmito tvary.

Obr. 18. – Barevný displej – grafická část



Obr. 19. – Černobílý displej – grafická část

5.2 Ovládání černobílého displeje

Pro snadnější a přehlednější programování jsem využil v tomto projektu hlavičkový soubor, v kterém mám uloženy všechny kódy příkazů pro tento řadič, viz tabulka 5, které jsem si pojmenoval podle sebe, pro snazší orientaci v kódu. Nemusím si tedy pamatovat pro každý příkaz zadanou hodnotu, jak je uvedeno v tabulce, ale stačí mi znát daný text. Soubor obsahuje i názvy metod, které se zde využívají.

Jedna z nejdůležitějších metod, kterou tento displej obsahuje je metoda s názvem ***void Inicializace().*** Tato metoda provádí základní nastavení displeje pro nastavený font 8x8 pixelů. Je zde proveden potřebný reset řadiče, a nastavení výstupních portu mikropočítače. Dále zde nastavuji základní rozložení paměti řadiče, určuji textovou a grafickou část a počet znaků na jeden řádek. Potřebné hodnoty, které zde potřebují, mám uložené v hlavičkovém souboru černobílého displeje. V tomto souboru si volím velikost daného fontu, jakmile zadám jinou hodnotu než 8, což představuje font 8x8, tak volím automatický font 6x8. V daném souboru si nastavuji i počet znaku na jeden řádek, jak pro textový (hodnota Text_area) tak i pro grafický režim (hodnota Graphic_area), určují si zde i počáteční adresy přidělené k jednotlivému režimu řadiče, pro textovou část je to hodnota Text_home, pro grafickou část je to Graphic_home. Jakmile si vyberu font o velikosti 6x8, tak tuto funkci ukončím a zavolám obdobnou metodu ale pro font 6x8, která má název ***void Inicializace6()***. Tato metoda je úplně stejná jako již zmíněna *Inicializace()*, akorát jsou zde jiné dosazované hodnoty pro tento font.

```
/*nastaveni fontu*/
#define Font_width          8
/*nastavena sirka displeje na 30znaku*/
#define Graphic_area        30
#define Text_area           30
/*nastavena sirka displeje na 40znaku*/
#define Text_area40         40
#define Graphic_area40      40
/*pocatecni hodnoty pameti*/
#define Text_home           0x0000
#define Graphic_home        0x2000
```

Nyní následuje příklad nastavení počáteční adresy pro textovou část.

```
CheckStatus();
ZapisData(Text_home & 0xff);
CheckStatus();
ZapisData(Text_home >> 8);
CheckStatus();
ZapisPrikaz(Set_text_home_address);
```

Základními metodami, bez kterých by displej nefungoval a nekomunikoval by s mikroprocesorem, jsou metody, v kterých posíláme řadiči různé zprávy neboli data. První metoda, kterou zde uvedu, slouží k posílání dat řadiči a jmenuje se ***void ZapisData(char data)*** se vstupním parametrem a to daty, které chceme poslat. Tato funkce funguje tak, že nejprve zkontrolujeme, zda je řadič připraven (viz kapitola 4.1.3). Následně nastavíme na řídicí pin displeje označovaného CD

logickou 0, tím řadiče přepneme do režimu, kdy bude pracovat s daty, nikoliv s instrukcemi. Pak na výstupní port mikropočítače vyšleme naše data, na řídicí piny CE a WR nastavíme logickou 0, tím aktivujeme, že řadič bude od tohoto okamžiku číst data. Následně vložíme zpoždění, aby měl řadič dostatek času přečíst si hodnoty z datové sběrnice, a na konec znovu nastavíme na řídicí piny WR, CE, CD původní hodnoty a to logické 1, abychom měli všechny řídicí signály, když je zrovna nevyužíváme v původních hodnotách jako po prvotní inicializaci.

```
void ZapisData (unsigned char dat)
{
  CheckStatus();
  LCD_CD = 0;
  LCD_Out = dat;
  LCD_CE = 0;
  LCD_WR = 0;
  delay();
  LCD_WR = 1;
  LCD_CE = 1;
  LCD_CD = 1;
}
```

Následující metoda slouží k posílání instrukcí neboli příkazu k řadiči, tato metoda se jmenuje ***void ZapisPrikaz (char prikaz)***. Řadič se ve výchozím stavu, po inicializaci rozhraní, nachází v režimu instrukcích, takže není zde třeba nastavovat pin CD na hodnotu 1. Vstupní parametr této funkce je náš požadovaný příkaz, který vyšleme na výstupní port mikroprocesoru. Nastavíme řadič, aby nyní četl data z datové sběrnice, toho docílíme tak, že na řídicí pin WR dáme logickou 0, tím deaktivujeme zápis, pin RD, který slouží jako signál pro čtení, je po inicializaci rozhraní v hodnotě logické 1, takže jej nemusíme tedy nastavovat. Musíme ale nastavit pin CE na logickou 0, abychom aktivovali řadič. Opět vložíme zpoždění, aby řadič měl dostatek času na přečtení daného příkazu. A na konci funkce nastavíme opět piny WR a CE na výchozí hodnoty logické 1.

```
void ZapisPrikaz (unsigned char prikaz)
{
  CheckStatus();
  LCD_Out = prikaz;
  LCD_WR = 0;
  LCD_CE = 0;
  delay();
  LCD_WR = 1;
  LCD_CE = 1;
}
```

Pro využití auto režimu řadiče (viz kapitola 4.1.3), který budu dále potřebovat při psaní textů, jsem naimplementoval následující tři metody. První metoda má název ***void NastaveniAutoRezimu()*** a jak již z názvu vyplývá, zabývá se tato metoda nastavením tohoto režimu, tato metoda obsahuje pouze příkaz „nastavení automatického zápisu“ (viz tabulka 5), který pošleme řadiči. Tento příkaz na datovou sběrnici pošleme jako hodnotu 0xB0,ale abychom si tuto

hodnotu nemuseli pamatovat, tak pod tuto hodnotou se skrývá v hlavičkovém souborů výraz Set_data_auto_write. Druhá metoda zabývající se auto režimem je metoda s názvem ***void ZapisDataAuto(char data).*** Tato metoda je obdobná s metodou pro klasické posílání dat, ale v této metodě voláme kontrolu stavu řadiče pro auto režim, nikoliv pro klasický. Třetí a poslední metodu, která se zabývá auto režimem je metoda nazvaná ***void ZrusAutoRezim()***. V této metodě posíláme řadiči opět jenom jeden příkaz a to „zrušení auto režimu“ (viz tabulka 5), pro kterou mám opět pojmenovanou v hlavičkovém souboru jako Auto_reset.

Metoda s názvem ***void TextGoTo(int x, int y)*** nám slouží pro převod zadaných souřadnic na jedinečnou adresu, podle které jsou následující data uložena do paměti řadiče. Tuto adresu si určíme podle následujícího vzorce:

*adresa = počáteční_adresa_textů + x + (počet_znaků * y)*

Tímto způsobem zajistíme, že pro každou souřadnici existuje právě jedno místo v paměti a nedojde tedy k přepsání již uložených hodnot. Jakmile máme tuto adresu vypočtenou, tak ji pošleme do další metody, která podle této adresy nastaví řadiči ukazatel do paměti právě na toto místo. Tato zmíněna metoda se nazývá ***void SetAddressPointer(char adresa)***, která právě využívá metody pro posílání dat *ZapisData* a posílá právě tuto adresu. Nakonec s využitím metody *ZapisPrikaz* pošle řadiči instrukci, aby nastavil ukazatele na tuto adresu, opět tento příkaz mám pojmenovaný v hlavičkovém souborů, jako Set_address_pointer.

Jakmile máme nastavenou adresu, kam se budou data ukládat, využijeme metodu nazvanou ***void lcd_retezec(char* text)***. Vstupním parametrem této funkce je libovolný textový řetězec. Pro psaní textu zde využívám nastavení auto režimu, proto je nutné nejprve zavolat metodu *NastaveniAutoRezimu* (viz výše), která nám tento režim připraví. Nyní procházíme tento vstupní řetězec po jednotlivých znacích, od kterých odečítám hodnotu 32 a posílám je jako vstupní parametr do funkce *ZapisDataAuto*, která je pošle řadiči, a ten jej zobrazí na displej. Tento cyklus opakuji tak dlouho než projdu všechny znaky daného řetězce. Nakonec zruším auto režim, tím že zavolám metodu *ZrusAutoRezim*.

Další implementovaná metoda se nazývá ***char ReadData()***, ta nám slouží ke čtení dat z řadiče. Z výstupních portů mikroprocesorů v této funkci provedeme vstupní porty, přečteme hodnotu z řadiče a danou hodnotu si uložíme. Tato hodnota bude výstupní této funkce. Na konci opět nastavíme mikroprocesoru daný port na výstupní.

```
unsigned char ReadData (void)
{
 uint8_t tmp;
 CheckStatus();
 TrisD = 0xff;
 LCD_RD = 0;
 LCD_CE = 0;
 LCD_CD = 0;
 delay();
 tmp = LCD_Out;
 LCD_RD = 1;
 LCD_CE = 1;
```

```
 LCD_CD = 1;
 TrisD = 0x00;
 return tmp;
}
```

Následující metoda ***void SetPixel (int x, int y, int zobraz)*** je využívaná když pracujeme s grafickým režimem řadiče, a chceme zobrazit jednotlivý pixel displeje. Podle vstupní proměnné *zobraz*, zajišťujeme, zda chceme na daných souřadnicích pixel zobrazit nebo ne. Pro hodnotu zobraz=0 tento pixel nezobrazíme. Tato funkce je využívaná v kreslení úseček a kruhu. V této metodě si nejprve vypočteme adresu, podle které budou data uložená v paměti, tuto adresu si určíme podle zadaných souřadnic, kde k počáteční grafické adrese přičteme podíl X-ové souřadnice k velikosti fontů a součin souřadnice Y s počtem znaků na jeden řádek. Následně na tuto adresu nastavíme ukazatel pamětí, využijeme metodu *SetAddressPointer*. Nyní musíme poslat řadiči instrukci, aby řadič při čtení dat, daný ukazatel neměnil svou pozici, z důvodů potřeby přečíst z dané nastavené pozice hodnotu, která je zde zapsána. Jakmile získáme tuto hodnotu, tak musíme nastavit v této hodnotě příslušný bit na 1, jestliže jej chceme zobrazit, v opačném případě vložíme 0.

```
 unsigned char tmp;
 unsigned int address;

 address = Graphic_home + (x/Font_width) + (Graphic_area * y);
 SetAddressPointer(address);
 ZapisPrikaz(Data_read_and_nonvariable);
 tmp = ReadData();
 CheckStatus();
 if(zobraz)
 {
  tmp |= (1 << (Font_width - 1 - (x % Font_width)));
 } else
 {
  tmp &= ~(1 << (Font_width - 1 - (x % Font_width)));
 }
 WriteDisplayData(tmp);
```

Metoda ***void WriteDisplayData (char data)*** posílá vstupní data metodě *ZapisData*, ale s tím rozdílem, že jakmile se data uloží v řadiči, tak manuálně pošleme řadiči příkaz, pomoci metody *ZapisPrikaz*, aby zvýšil ukazatel do pamětí o 1, v mém případě mám tento příkaz uložený pod názvem Data_write_and_increment v hlavičkovém souborů. Této metody využívám k vymazání pamětí řadiče, které jsou implementovány v metodách ***void ClearText(void)***, ***void ClearCG(void)*** a ***void ClearGraphic(void)***.Tyto tři metody jsou hodně podobné i když každá tato metoda je určená pro mazání jiné části pamětí, *ClearText* je určena pro mazání textové části pamětí, *ClearCG* je určena pro mazání vytvořených znaků a *ClearGraphic* je určená pro mazání grafické části pamětí. Na začátku každé z těchto metod je nastaven ukazatel pamětí na začátek této části, toho je docíleno použitím metody *SetAddressPointer*. Pomoci cyklu, který je nastaven na jednotlivé počátky daných pamětí, až po jejich maximální hodnoty, do kterých mohou ukládat svá

data. Procházíme nyní jednotlivé části pamětí a zapisujeme do nich, díky již zmiňované metody *WriteDisplayData*, samé nuly, tím paměť vymažeme.

Metoda ***void kurzor_xy (char x, char y)*** nastaví kurzor na zadanou polohu displeje, podle souřadnic x, y. Tato metoda pošle řadiči nejprve tyto dvě hodnoty, které pro nás představují souřadnice, a následně pošle příkaz nastavení ukazatele na kurzor, díky kterému řadič vykreslí kurzor na displej.

Metoda s názvem ***void posunDolu (int cislo)*** nám podle vstupního parametru nastaví o kolik řádků má řadič posunout základní textovou část směrem dolů (viz kapitola 4.1.9). Tuto metodu využívám u rolování textů, jak již zde bylo popsáno.

Následující dvě metody slouží k stránkování displeje, jejich princip spočívá v nastavování různých počátku zobrazovaných dat. Pojmenovaná metoda ***void lcd_xy_stranka (int x, int y, int stranka)*** má stejný význam jako metoda *TextGoTo()*, tedy nastavuje nám podle zadaných souřadnic ukazatel do pamětí, v tomto případě ještě podle vstupního parametru stranka určuje v které části paměti budou data uložena, tj. na které stránce se nám například zobrazí text, když po této metodě zavoláme metodu *lcd_retezec()*. Následující metoda zvaná ***void lcd_zmenStranku(int stranka)*** nám slouží pro přepínání mezi jednotlivými stránky.

5.3 Ovládání barevného displeje

U barevného displeje jsem taktéž použil v tomto projektu hlavičkový soubor, v kterém jsou vypsané všechny metody, které se zde využívají. Ale naproti černobílého displeje, kde mám vypsané všechny příkazy k řadiči a podle sebe tyto příkazy přejmenované, tak zde nemám všechny příkazy vypsané a přejmenované, z důvodů, že se zde vyskytuje o mnoho víc příkazu, a z důvodů že většinu těchto příkazu ani nevyužívám. Proto jsem se rozhodl, že jednotlivé příkazy v programu, které budu zrovna potřebovat budu zadávat přímo podle hodnot udávaných v dokumentaci řadiče, tj. budu zadávat hexadecimální hodnoty. V hlavičkových souborech mám předdefinované a vytvořené znaky dvou velikostí, které zde budu využívat.

V hlavním zdrojovém souboru tohoto displeje (TFT_LCD.c) mám nadeklarované čtyři globální proměnné. Tyto proměnné využívám v různých metodách a slouží mi k nastavení různých hodnot, podle kterých provádím příslušné operace. Jedná se o proměnnou nazvanou *barva*, která je typu int, a podle této hodnoty, kterou ji zadáme, pak nastavuji počet zobrazovaných barev na displeji (viz kapitola 2.5.1.1), v případě, že zadáme hodnotu 0, tak řadič bude pracovat se všemi barvy. V případě, že zadáme jinou hodnotu, tak řadič bude pracovat pouze v režimu 8 barev. Taktéž pro proměnnou *barevny_filtr* jestliže ji zadáme hodnotu 1, tak řadič bude používat RGB filtr, pro jinou hodnotu budeme používat BGR filtr (viz kapitola 4.2.2). S proměnnou *otoceni* nastavujeme úhel otočení, které může nabývat hodnot 0, 90, 180 a 270 jiné hodnoty zde nejsou přípustné. Poslední globální proměnná je *scroll*, jestliže ji přiřadíme hodnotu 0, tak vypneme rolování textu, a pro 1 zapneme.

Nejdůležitější metodou je zde ***void main_init(void)***, která slouží k nastavení řadiče. V této metodě volám nejdříve metodu ***void rozhrani()***, v které mám nastavené výstupní porty mikroprocesoru, aby mikroprocesor věděl, které porty má využívat. Následuje provedeny reset řadiče. V další části této funkce je série nastavování jednotlivých hodnot řadiči, jako je nastavení frekvence oscilátoru, gamma, nastavení otočení, zapínaní nebo vypínaní rolování, určování počtu

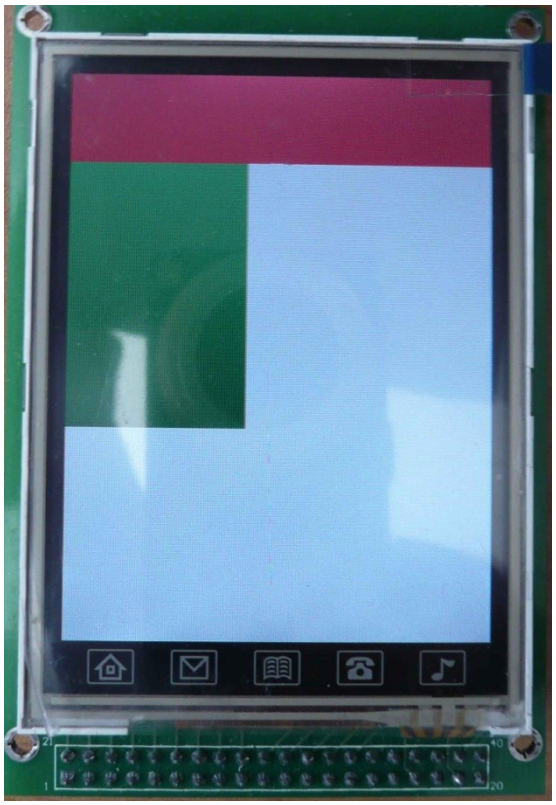
barev s kterými bude řadič pracovat, filtr zobrazení a na konci volám další metodu, kterou mám pojmenovanou ***void zapniDisplej(void)*** v které zapínám displej. Ve všech těchto případech využívám pro nastavování těchto hodnot metody ***void main_W_com_data()***, kterou ještě později popíši.

Další velmi důležitou metodou je ***void main_W_com_data(int prikaz, int data)***, která nejprve posílá daný příkaz do metody ***void main_Write_COM(int prikaz)***, která tento příkaz pošle na datovou sběrnici. V tomto projektu používám, jak již zde bylo zmíněno, pro komunikaci mezi mikroprocesorem a displejem 16-bitovou datovou sběrnici, proto na port D mikroprocesoru posílám horní část byte (bity 15-8) a na port C dolní část byte (bity 7-0). Druhá metoda, která se zde nachází se jmenuje ***void main_Write_DATA(int data),*** která posílá na datovou sběrnici data se stejným principem jako v metodě *main_Write_COM*. Následuje ukázka kódu z metody main_Write_COM.

```
LCD_RS = 1;
LCD_CS = 0;
LCD_Out2 = (DH >> 8);
LCD_Out1 = (DH & 0x00FF);
LCD_WR = 0;
_asm
  nop
_endasm;
LCD_WR = 1;
LCD_CS = 1;
```

Metody pojmenované ***void zobrazeni320x240*** a ***void zobrazeni240x320***, jsou metody, které volám z metody *main_init(void)* a slouží k nadefinování velikosti zobrazované plochy. Obě tyto metody jsou totožné, až na to, že jednu používám pro otočení 0 a 180 stupňů a druhou pro 90 a 270 stupňů otočení. V každé z těchto metod jsou jinak nastavené počátky a konce řádků a sloupců.

Metody ***void clear240x320(int barva)*** a ***void clear320x240(int barva)*** jsou metody, které slouží k nastavení dané barvy na celý displej, každá metoda je určena pro jiné otočení displeje, jak již z názvu vyplývá, *clear240x320* je pro otočení 0 nebo 180 stupňů, *clear320x240* je pro otočení 320x240 stupňů. Princip těchto metod spočívá, že jsi nejprve nastavíme pomoci metody address_set rozsah pamětí v které budeme provádět změny, tzn. rozsah bude nastaven od počátku [0, 0] a bude končit [239, 319], respektive [319, 239] podle úhlů otočení. Následně s využitím dvou cyklů procházíme každý jednotlivý pixel a pomoci metody *main_Write_DATA* nastavujeme jednotlivým pixelům barvu. Obdobná metoda se nazývá ***BarvaPozadi(int x1, int y1, int x2, int y2, int barva)***, která pracuje na stejném principu, akorát zde volíme barvu pozadí s určitými souřadnicemi, ne tedy na celý displej. Při nastavení metody BarvaPozadi(0, 0, 239, 50, cervena) a BarvaPozadi(0, 51, 100, 200, zelena) získáme následující obrázek.



Obr. 20. – Barevné pozadí displeje

S metodou ***void address_set(int x1, int y1, int x2, int y2)*** nastavujeme podle zadaných souřadnic řadiči oblast paměti, s kterou budeme pracovat. V této metodě využíváme metody *main_W_com_data*, a tuto oblast určujeme pomoci registru pro určení velikosti displeje (příkazy 0x02 až 0x09).

Velmi důležitou metodou je metoda pro psaní textu. Tato metoda se nazývá ***void napisRetezec(int x, int y, char* text, int barvaPisma, int barvaPozadi)***, která využívá pro psaní malý font a metoda ***void napisRetezecVelky(int x, int y, char* text, int barvaPisma, int barvaPozadi)***, která využívá pro psaní velký font. V těchto metodách jsou navíc oproti černobílého displeje, souřadnice s kterými určujeme pozici textu. Těla těchto metod se skládá z cyklu, v kterém postupujeme po jednotlivých znacích našeho daného řetězce a posíláme tyto znaky metodě *napisZnak*, pro malý font, nebo *napisZnakVelky*, pro velký font, které je postupně zobrazují na displej. V tomto cyklu, jakmile pošleme daný znak, tak musíme k X souřadnici přičíst šířku jednotlivých fontů, tedy pro malý font přičteme 7, pro velký font přičteme 15. V případě, že nám X souřadnice přesáhne šířku displeje, tak tuto souřadnici vynulujeme a k souřadnici Y přičteme 1.

Metoda ***void napisZnak(int x, int y, char znak, int barvaPisma, int barvaPozadi)*** je již zmiňovaná metoda, která posílá jednotlivé znaky na displej. Využíváme metody *address_set*, do které zadáváme souřadnice počátků X a Y a koncové souřadnice podle velikosti fontů, tedy pro malý font přičteme k X-ové souřadnici 7 a k Y-ové souřadnici 11. Pro velký font jsou tyto hodnoty jiné, k X-ové souřadnici přičteme 15 a k Y-ové souřadnici 23. Následuje operace, v které odečteme od požadovaného znaku hodnotu 32 (viz kapitola 4.2.5) a následně tuto hodnotu ještě vynásobíme pro malý font 12 a pro velký font hodnotou 48 a tím dostaneme první hodnotu našeho znaku.

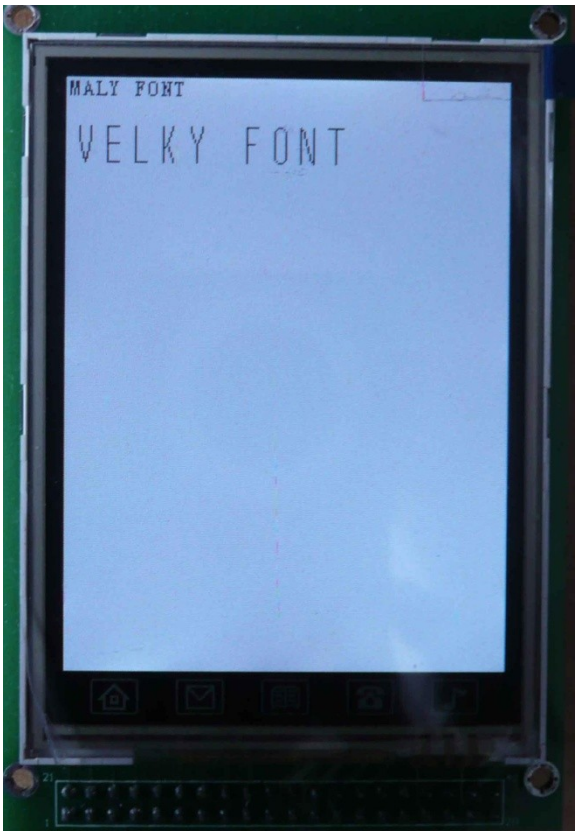
Násobíme z důvodu, protože jednotlivé vytvořené znaky se skládají z 12, respektive 48 hexadecimální hodnot, v kterých je nadefinován jejich tvar. Poté postupně procházíme hodnoty vytvořeného znaku a po jednotlivých bitech tento znak vykreslujeme na displej.

```
unsigned char *temp = znaky;
address_set (x, y, x+7, y+11);
temp += (value-32)*12;
for(j=0; j<12; j++)
{
        for(i=0;i<8;i++)
        {
                if((*temp&(1<<(7-i)))!=0)
                {
                        main_Write_DATA(dcolor);
                }
                else
                {
                        main_Write_DATA(bgcolor);
                }
        }
        temp++;
}
```

V tomto projektu jsou vytvořené i dvě pomocné metody, do kterých zadáváme pouze číslo řádků, a následně nám tyto metody vrátí příslušné souřadnice onoho řádků. Tyto metody se nazývají ***int radek(int cislo)*** a ***int radekVelky(int cislo)*** tyto metody se dají využívat v případě psaní textu.

```
int radek (int cislo)
{
  int pozice_radku = 0;
  pozice_radku = cislo * 12;
  return pozice_radku;
}
```

Jestliže využijeme tyto metody s těmito vstupními parametry, napisRetezec(2, radek(0), „MALY FONT“, cerna, bila) a napisRetezecVelky(2, radekVelky(1), „VELKY FONT“, cerna, bila) získáme na displeji tyto znaky, viz následující obrázek.



Obr. 21. – Psaní textu, barevný displej

6. Praktická realizace

Abychom si mohli vyzkoušet praktickou realizaci práce s displejem a vyzkoušení si některých vlastností daných displejů, tak musíme postupovat následujícím způsobem. Nejprve musíme v obou případech jak pro černobílý tak i barevný displej nastavit příslušné porty mikroprocesoru jako výstupní. Výstupní port mikroprocesoru nastavíme tak, že na pomocný řídící registr daného portu (TRIS registr) nastavíme 0, v opačném případě když nastavíme řídící registr daného portu na 1 tak docílíme toho, že daný port bude sloužit jako vstup do mikroprocesoru.

6.1 Černobílý displej

Jakmile máme nastavené vstupní a výstupní porty mikroprocesoru, tak je doporučováno provést reset řadiče alespoň po dobu 1ms, toho dosáhneme vysláním logické 1 na pin RST displeje, tímto signálem řadič vypneme. Nyní počkáme alespoň 1ms a opět znovu nastavíme pin RST ale tentokrát na hodnotu logické 0, tímto signálem opět řadič zapneme.

6.1.1 Počáteční adresy

Po resetu nastavíme velikost fontů (viz kapitola 4.1.2) a následně můžeme posílat řadiči data ohledně přiřazení počátečních adres paměťového prostoru pro textovou a grafickou část (viz kapitola 4.1.1). Za každým nastavením počátečních adres musíme řadiči poslat příslušný příkaz, aby řadič věděl, jaké data má ukládat od dané hodnoty. Dalším důležitým nastavením je určení počtu zobrazovaných znaku na jeden řádek (viz kapitola 4.1.2) a následně opět zaslání příkazu pro potvrzení zadaných hodnot.

6.1.2 Režim zobrazení

Poslední příkazy, které pošleme řadiči je nastavení režimu zobrazení, jestli budeme zobrazovat data z textové části paměti nebo z grafické části paměti, posledním příkazem je určení režimu zobrazení (viz kapitola 4.1.5). Po tomto dokončení máme úspěšně nastaveny řadič.

6.1.3 Psaní textů

Při psaní libovolného textu na displej musíme nejprve nastavit řadiči, aby ukládal data do své pamětí a to přesněji do té části, kterou jsme nastavili pro text (viz kapitola 4.1.6). Toto nastavení provedeme tak, že pošleme řadiči počáteční adresu textové části, a následným příkazem 0x24, což nám představuje nastavení adres pointerů.

Pro snazší práci s textem je doporučené nastavení auto režimu řadiče (viz kapitola 4.1.6). Jakmile máme nastaveny tento auto režim, tak procházíme náš zadaný text po jednotlivých znacích a každý jednotlivý znak zasíláme řadiči a ten jej zobrazí na displeji. Nesmíme však zapomenout při posílání znaků od každého odečíst hodnotu 32, protože jak již zde bylo zmíněno tak řadič obsahuje znaky z ASCII tabulky od hodnoty 32. Pro zjednodušení si můžeme představit, že řadič obsahuje předem definované znaky, které jsou uloženy v jednom poli. Nyní když požadujeme zapsat například znak A na displej, jenž má hodnotu podle ASCII tabulky 65, tak musíme odečíst již několikrát zmíněnou hodnotu 32, neboť v našem případě v řadiči má znak A hodnotu 33. Jakmile tímto způsobem projdeme celý náš zadaný řetězec, tak nesmíme zapomenout zrušit auto režim řadiče.

6.1.4 Zobrazování grafických obrazců

Při zobrazování grafických tvarů postupujeme podobně jako v postupu pro psaní textu (kapitola 5.1.3) akorát zde musíme nastavit řadiči počáteční adresu pro grafickou část, aby řadič ukládal data od této adresy.

S využitím znalostí zobrazovacích algoritmů, můžeme zobrazit námi požadované geometrické tvary, viz kapitola 4.3.

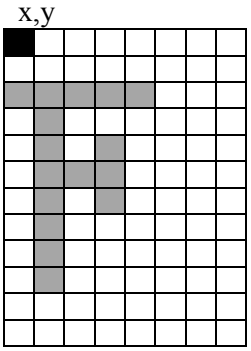
6.2 Barevný displej

V situaci, jestliže máme nastavené vstupní a výstupní porty mikroprocesoru, musíme nejprve provést reset řadiče jako v případě černobílého displeje, to provedeme tak, že na pin REST displeje pošleme logickou 0, tím řadič vypneme a po uplynutí alespoň 10ms na tento pin opět nastavíme logickou 1, s čímž opět zapneme řadič.

Po resetu řadiče nastavíme velikost zobrazovaného displeje (viz kapitola 4.2.1). Musíme taky řadiči nastavit režim zobrazení, musíme zvolit mód řadiče (viz kapitola 2.5.1.1), režim otočení (viz kapitola 4.2.3), nastavit frekvenci řadiče a v neposlední řadě nastavení úrovně gamma.

6.2.1 Psaní textů

Po úspěšném dokončení inicializace displeje, se nyní zaměříme na psaní textů. Využijeme naše vytvořené znaky, které máme uložené v mikroprocesoru (viz kapitola 4.2.5). Umisťování textu na displej probíhá pomoci námi zvolené souřadnice, která představuje levý horní pixel prvního znaku textu.



Obr. 22. – Souřadnice znaků

Pomoci cyklu procházíme jednotlivé znaky zadaného řetězce, a to do té doby než dojdeme na konec. V každém průchodu tohoto cyklu u každého znak musíme upravovat jejich souřadnice, aby se jednotlivé znaky navzájem nepřepisovaly. Tato úprava spočívá v tom, že jakmile pošleme řadiči první znak i se souřadnicí, tak musíme přičíst k souřadnici X takovou hodnotu, jakou nám představuje šířka našeho vytvořeného znaku v pixelech. Analogický toto platí i pro souřadnici Y, kde musíme kontrolovat, jestliže nám souřadnice X nepřesáhla hodnotu 239, což je hranice našeho displeje. Jestliže se toto stalo, musíme k souřadnici Y přičíst takovou hodnotu, která nám vyjadřuje výšku našeho vytvořeného znaku v pixelech. Každý znak je tedy jednotlivě posílán řadiči s jeho unikátní souřadnicí.

6.2.2 Zobrazování grafických obrazců

Zobrazování grafických obrazců funguje na principu, v kterém postupujeme po jednotlivých pixelech displeje a zobrazujeme je. Při tomto zobrazování jednotlivých pixelů

využíváme již zmíněné registry v řadiči, které slouží k určování velikosti zobrazované plochy (viz kapitola 4.2.1), v tomto případě zadáme začátek sloupce na požadovaný pixel (souřadnice X), konec sloupce na hodnotu o 1 větší než začátek, začátek řádků nastavíme podle souřadnice Y a konec řádků opět o 1 větší než začátek řádků. Tímto způsobem provedeme to, že požadovaný pixel se rozsvítí.

S využitím tohoto postupu a znalosti zobrazovacích algoritmů (viz kapitola 4.3) můžeme snadno zobrazit požadované geometrické tvary na displej.

7. Porovnání naměřených hodnot

Pro případné použití těchto displejů v praxi jsem následně provedl několik měření, v kterých jsem se zaměřil na měření vstupního proudu, abych zjistil, jaké mají tyto displeje požadavky na zdroj elektrického proudu, v případě jejich použití. Tyto moje naměřené hodnoty jsem uvedl v následující tabulce.

7.1 Naměřené hodnoty

měřena veličina	černobílý displej	barevný displej
vstupní napětí displeje	+5 V	+5 V
napětí podsvětlení	+5 V	+2,7 V
odběr proudu – podsvětlení	140 mA	11,4 mA
odběr proudu	290 mA	

Tabulka 9 – Naměřené hodnoty

Z naměřených hodnot vyplývá, že barevný displej je méně náročný na zdroj elektrického proudu než černobílý. Využití černobílého displeje je tedy značně omezené v použití v mobilních zařízeních, kde požadujeme určitou výdrž na baterii. Naproti tomu, barevný displej je k takovým to účelům naprosto vhodný. Černobílý displej je naopak vhodný k použití například v průmyslových terminálech, kde nepožadujeme barevné rozhraní.

7.2 Zkoumání rychlosti rolování textu

Při dalším zkoumání vlastností těchto displejů jsem se zaměřil na porovnání rychlosti rolování textu u jednotlivých displejů, protože předpokládám, že oba displeje budou následně využity v různých aplikacích, kde bude rolování textu či grafiky používáno.

Při zkoumání rychlosti rolování textu jsem zjistil, že rychlost mikroprocesoru nemá příliš velký vliv na tento čas. Rychlost rolování textu je totiž závislá pouze na rychlosti daného řadiče displeje a nikoliv na rychlosti mikroprocesoru nebo šířky jeho datové sběrnice. Frekvence mikroprocesoru nám ovšem ovlivňuje rychlost, kterou jsou řadiči posílaná data. Při vyšší frekvenci mikroprocesoru máme rychlejší odezvu od řadiče. Z důvodu, že data jsou mikroprocesorem posílaná rychleji.

LCD displej	počet řádků	použitý krystal mikroprocesoru	čas
černobílý displej (PQG2412AW-SYL)	100	40 MHz	0,8s
barevný displej (TFT 320QVT)	100	40 MHz	0,4s

Tabulka 10 – Rolování textu

8. Závěr

Cíle práce byly splněny podle zadání. Na základě předem získaných informací a požadavku této práce jsem se rozhodl o pořízení právě těchto dvou displejů, černobílého PQG2412AW-SYL a barevného TFT 320QVT na kterých byla tato práce předváděna a popisovány funkce jednotlivých řadičů. Na nepájivé desce plošných spojů byla provedená realizace propojení mikroprocesoru a daného LCD displeje, včetně vytvořeného programátoru mikroprocesoru. Všechny zde uvedené příklady byly vyzkoušeny na daných displejích a během testování nebyly zjištěny žádné závažné chyby ve funkčnosti. Objevil jsem akorát dvě chyby, které se mi bohužel nepovedly odstranit. Jedna chyba nastává někdy u černobílého displeje, když odpojím programátor od PC, tak se mi na displeji někdy zobrazují náhodné znaky. A druhá chyba nastává u barevného displeje u dotykové obrazovky, i když tato funkce nebyla nijak implementována, tak v situaci když mám na displeji nějaké tvary nebo text a přejedu prstem po této obrazovce, tak mi zmizí veškeré znaky a text z obrazovky.

Součástí této práce je i vypracovaná programová část pro mikroprocesor, jenž slouží k řízení daných displejů. Toto programové řešení jsem vytvořil ve vývojovém prostředí Piklab s využitím kompilátorů SDCC v operačním systému Linux.

9. Seznam použité literatury

[1] T6963C [PDF dokument].
Dostupné na <http://www.datasheetcatalog.org/datasheet/toshiba/1908.pdf>

[2] HX8347-A [PDF dokument].
Dostupné na <http://www.datasheet4u.net/share_search.php?sWord=HX8347-A>

[3] PIC18F452 [PDF dokument].
Dostupné na <http://www.datasheetcatalog.org/datasheet/microchip/39564b.pdf>

[4] Horák, Jaroslav. *Hardware učebnice pro pokročilé 3.aktualizované vydání.* Computer Press, 2005, 344 s, ISBN 80-251-0647-0

[5] Vacek, Václav Ing. Vlček, Jiří Ing. *Praktické využití procesoru PIC*, Ben, Praha 2001, 72 s

[6] Vacek, Václav. *Učebnice programování PIC*, 1. vydání, Ben, Praha 2000, 139 s, ISBN 80-86056-87-2

[7] Kučera, Jan. *Grafické LCD displeje a jejich řízení*, 19.červen 2002
Dostupné na <http://hw.cz/Firemni-clanky/Elatec/ART309-Graficke-LCD-displeje-a-jejich-rizeni.html>

[8] Bresenham's line algorithm.
Dostupné na <http://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm>

10. Seznam příloh

Příloha 1: *Seznam použitých součástek*

Rezistory:

R1, R2, R4	10k
R3, R5	100k
R6	47k
R7, R8, R9, R10	2k2
R11	1k

Kondenzátory:

C1, C2	1μF

Tranzistory:

T1, T2, T3	BC547
Q1	BC557

Integrované obvody:

IO1	4049N
IO2	7805T
IO3	7812T

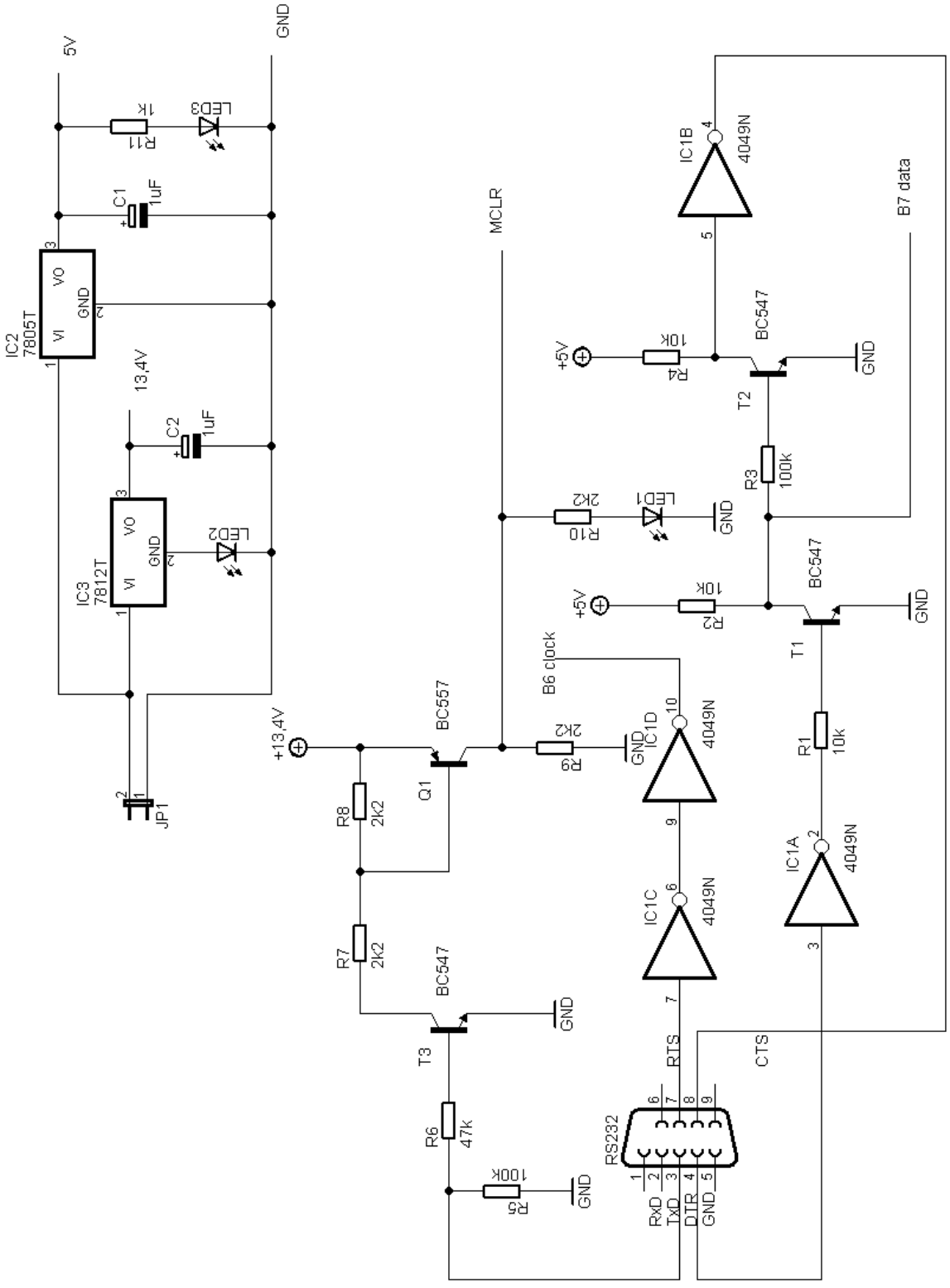
Diody:

LED1	LED červená 5mm
LED2, LED3	LED zelená 5mm

Konektory:

RS232	FD9(Canon 9 pinů) s volným koncem

Příloha 2: *Schéma zapojení programátoru*



Příloha 3: *ASCII tabulka*

kód	znak	kód	znak	kód	znak	kód	znak	kód	znak
0032	mezera	0077	**M**	0122	**z**	0167	**§**	0212	**Ô**
0033	**!**	0078	**N**	0123	**{**	0168	**¨**	0213	**Ő**
0034	**"**	0079	**O**	0124	**|**	0169	**©**	0214	**Ö**
0035	**#**	0080	**P**	0125	**}**	0170	**Ş**	0215	**×**
0036	**$**	0081	**Q**	0126	**~**	0171	**"**	0216	**Ř**
0037	**%**	0082	**R**	0127	**□**	0172		0217	**Ů**
0038	**&**	0083	**S**	0128	**€**	0173		0218	**Ú**
0039	**,**	0084	**T**	0129	□	0174	**®**	0219	**Ű**
0040	**(**	0085	**U**	0130	**,**	0175	**Ż**	0220	**Ü**
0041	**)**	0086	**V**	0131	□	0176	**°**	0221	**Ý**
0042	*****	0087	**W**	0132	**"**	0177	**±**	0222	**Ţ**
0043	**+**	0088	**X**	0133	**…**	0178	**˛**	0223	**ß**
0044	**,**	0089	**Y**	0134	**†**	0179	**ł**	0224	**ŕ**
0045	**-**	0090	**Z**	0135	**‡**	0180	**´**	0225	**á**
0046	**.**	0091	**[**	0136	□	0181	**µ**	0226	**â**
0047	**/**	0092	****	0137	**‰**	0182		0227	**ă**
0048	**0**	0093	**]**	0138	**Š**	0183	**·**	0228	**ä**
0049	**1**	0094	**^**	0139	**‹**	0184	**¸**	0229	**Í**
0050	**2**	0095	**_**	0140	**Ś**	0185	**ą**	0230	**ć**
0051	**3**	0096	**`**	0141	**Ť**	0186	**ş**	0231	**ç**
0052	**4**	0097	**a**	0142	**Ž**	0187	**"**	0232	**č**
0053	**5**	0098	**b**	0143	**?**	0188	**Ľ**	0233	**é**
0054	**6**	0099	**c**	0144	□	0189	**˝**	0234	**ę**
0055	**7**	0100	**d**	0145	**'**	0190	**ľ**	0235	**ë**
0056	**8**	0101	**e**	0146	**'**	0191	**ż**	0236	**ě**
0057	**9**	0102	**f**	0147	**"**	0192	**Ŕ**	0237	**í**
0058	**:**	0103	**g**	0148	**"**	0193	**Á**	0238	**î**
0059	**;**	0104	**h**	0149	**o**	0194	**Â**	0239	**ď**
0060	**<**	0105	**i**	0150	**-**	0195	**Ă**	0240	**đ**
0061	**=**	0106	**j**	0151	**-**	0196	**Ä**	0241	**ń**
0062	**>**	0107	**k**	0152	□	0197	**Ĺ**	0242	**ň**
0063	**?**	0108	**l**	0153	**™**	0198	**Ć**	0243	**ó**
0064	**@**	0109	**m**	0154	**š**	0199	**Ç**	0244	**ô**
0065	**A**	0110	**n**	0155	**›**	0200	**Č**	0245	**ő**
0066	**B**	0111	**o**	0156	**ś**	0201	**É**	0246	**ö**
0067	**C**	0112	**p**	0157	**ť**	0202	**Ę**	0247	**÷**
0068	**D**	0113	**q**	0158	**ž**	0203	**Ë**	0248	**ř**
0069	**E**	0114	**r**	0159	**ź**	0204	**Ě**	0249	**ů**
0070	**F**	0115	**s**	0160	**t. mezera**	0205	**Í**	0250	**ú**
0071	**G**	0116	**t**	0161	**ˇ**	0206	**Î**	0251	**ű**
0072	**H**	0117	**u**	0162	**˘**	0207	**Ď**	0252	**ü**
0073	**I**	0118	**v**	0163	**Ł**	0208	**Đ**	0253	**ý**
0074	**J**	0119	**w**	0164	**¤**	0209	**Ń**	0254	**ţ**
0075	**K**	0120	**x**	0165	**Ą**	0210	**Ň**	0255	**˙**
0076	**L**	0121	**y**	0166	**¦**	0211	**Ó**		

Příloha 4: *Obsah přiloženého CD*

Přiložené CD obsahuje zdrojové kódy programu, text bakalářské práce ve formátu PDF, fotografie černobílého a barevného displeje a jejich videa. V následující tabulce je popsána struktura CD.

Adresář		Popis
cernobily displej		
	/kody	zdrojové kódy pro černobílý displej
	/obrazky	obrázky černobílého displeje
	/videa	videa černobílého displeje
barevny displej		
	/kody	zdrojové kódy pro barevný displej
	/obrazky	obrázky barevného displeje
	/videa	videa barevného displeje
text_klo126.pdf		text bakalářské práce